A CONCEPT FOR SURFACE RECONSTRUCTION FROM DIGITISED DATA.

Wolfgang Josef Schinke

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

DeMontfort University, Leicester, United Kingdom

September 2001

BRITISH LIBRARY

IMAGING SERVICES NORTH

Boston Spa, Wetherby
West Yorkshire, LS23 7BQ
www.bl.uk

THESIS CONTAINS MANY CDs/DVDs

-UNABLE TO COPY-

IF REQUIRED PLEASE CONTACT THE UNIVERSITY

© Copyright 2001
by
Wolfgang Josef Schinke

Abstract

Reverse engineering and in particular the reconstruction of surfaces from digitized data is an important task in industry. With the development of new digitizing technologies such as laser or photogrammetry, real objects can be measured or digitized quickly and cost effectively. The result of the digitizing process is a set of discrete 3D sample points. These sample points have to be converted into a mathematical, continuous surface description, which can be further processed in different computer applications. The main goal of this work is to develop a concept for such a computer aided surface generation tool, that supports the new scanning technologies and meets the requirements in industry towards such a product.

Therefore first, the requirements to be met by a surface reconstruction tool are determined. This marketing study has been done by analysing different departments of several companies. As a result, a catalogue of requirements is developed. The number of tasks and applications shows the importance of a fast and precise computer aided reconstruction tool in industry. The main result from the analysis is, that many important applications such as stereolithographie, copy milling etc. are based on triangular meshes or they are able to handle these polygonal surfaces.

Secondly the digitizer, currently available on the market and used in industry are analysed. Any scanning system has its strength and weaknesses. A typical problem in digitizing is, that some areas of a model cannot be digitized due to occlusion or obstruction. The systems are also different in terms of accuracy, flexibility etc. The analysis of the systems leads to a second catalogue of requirements and tasks, which have to be solved in order to provide a complete and effective software tool. The analysis also shows, that the reconstruction problem cannot be solved fully automatically due to many limitations of the scanning technologies.

Based on the two requirements, a concept for a software tool in order to process

digitized data is developed and presented. The concept is restricted to the generation of polygonal surfaces. It combines automatic processes, such as the generation of triangular meshes from digitized data, as well as user interactive tools such as the reconstruction of sharp corners or the compensation of the scanning probe radius in tactile measured data.

The most difficult problem in this reconstruction process is the automatic generation of a surface from discrete measured sample points. Hence, an algorithm for generating triangular meshes from digitized data has been developed. The algorithm is based on the principle of *multiple view combination*. The proposed approach is able to handle large numbers of data points (examples with up to 20 million data points were processed). Two pre-processing algorithm for triangle decimation and surface smoothing are also presented and part of the mesh generation process. Several practical examples, which show the effectiveness, robustness and reliability of the algorithm are presented.

Acknowledgements

First of all I would like to thank my wife Natalia for her patience, understanding and unconditional support during all the years that I have worked on my PhD. There were probably more agreeable subjects to hear weekends and other social activities beeing cancelled or postponed because of professional and PhD priorities.

I would also like to thank Prof. Jonathan Blackledge, for his open mind and his non-conformist approach to scientific problems and the world in general. He immediately took interest in the proposed PhD and was very flexible and understanding when I made suggestions to redirect the work to the area of surface reconstruction.

I would like to show my gratitude to the company Tebis AG, especially to one of its owners, Mr. Bernhard Rindfleisch. He not only allowed me to do this PhD part-time besides daily work, but he also forced this work by finding a practically related subject and experienced project partners in industry. He also guided me through this work with all his knowledge and experience. Without him this work might not have been possible.

To Florian Albat from Tebis AG, mathematician and perfectionist in software development. He never gave up trying to teach me the principles in software development and mathematics even if it often seemed to be a hopeless task.

Also I would like to thank Willi Riep from Tebis AG for all the fruitful discussions and ideas related to this work. I am especially grateful for his ability and patience to find and solve bugs in other peoples software and also for reading and commenting on my thesis.

I would like to thank my colleague Joachim Schuster, who had the idea of continuing research and who encouraged me to pursue a doctoral degree. I am also grateful to him for reading and commenting on my thesis.

I would also like to thank all industrial project partners, to many to mention all

by name, helping me to understand the industrial demanding aspects of this project and supporting me with all their practical knowledge.

To the staff of DeMontfort University, who have allowed me to continue my PhD which had been started at Cranfield University.

Finally to my children Patrick and Sophia who never gave up showing me that there is life besides surface reconstruction.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Specification of the Problem

The development of a new product such as a car-body or the interior of a car is a very time consuming process. Modern computer technologies can reduce the amount of development time and costs in specific areas of this process. Deformation tests e.g. using *virtual* computer models and finite element analysis can reduce the number of analyses to be done with *real* physical models. Automatic toolpath generation systems in combination with numerically controlled machine tools can quickly generate real models from Computer Aided Design (CAD) surfaces. In comparison to old fashioned copy milling machines, this process needs only a small amount of time.

There are many tasks which can be solved faster and more cost effectively using computer systems and virtual models, but not everything can be done without having a real model. Of course at the end of the development process, a real model has to be manufactured. But also in early stages of the design and manufacturing, a physical model is sometimes required. For example, stylists still need real models to get a good impression on the shape, the functionality and the overall concept of a part. Today, there is no software system available, which can simulate these effects efficiently and sufficiently to a stylist.

Therefore for many reasons it is required to have both, a real and a virtual computer model. It is also necessary to convert one representation into the other one. Figure 1 illustrates the task.

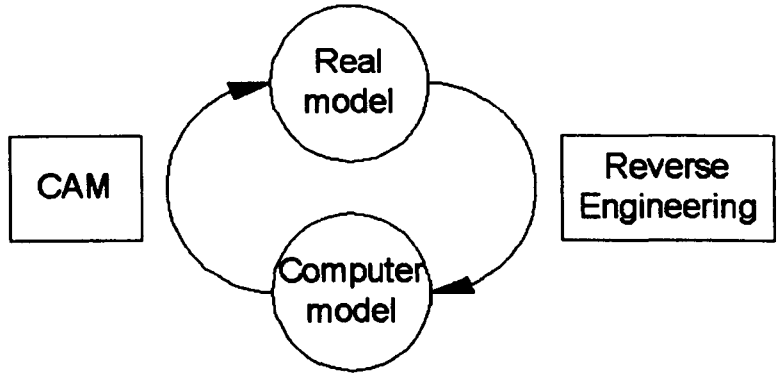


Figure 1: CAM and Reverse Engineering

The generation of physical parts from computer models is an application for Computer Aided Manufacturing (CAM) packages. The virtual surface of a computer model is usually described in continuous mathematical descriptions such as spline surfaces or solids. From these surface models, manufacturing data such as an numerically controlled (NC) toolpath is generated and can be further processed within NC manufacturing machines.

The generation of virtual models from real physical models is often called a *reverse engineering* task. Within reverse engineering tools, mathematical surfaces are generated from the part dimensions. One possibility of how to obtain these part dimensions is to measure the surface of a real part at certain sample points, from which a continuous surface can be estimated. There are different measuring machines available for the generation of these sample points. *Tactile* measuring machines having available on the market for more than 30 years. Optical measuring systems such as *laser* or *structured lighting* systems are relatively new and provide the user with faster and more cost effective methods for digitizing. The 3D sample points have to be converted into continuous mathematical surface descriptions such as polygonal or polynomial surface models. This *surface reconstruction* task is necessary for further processing within different applications since they are usually based on surfaces rather than on discrete sample points. Many applications such as toolpath generation, stereolithography, visualisation or finite element analysis are based on triangular meshes.

Few software systems are able to solve the tasks involved in this surface reconstruction process, but none of them provide methods which are accurate, effective and

easy enough to match all the requirements, which are demanded within this process. The reasons for this are multifaceted. Some of the problems arise from the limitations of the scanning systems. For example details smaller than the resolution of a scanner camera cannot be digitized. There are limitations in storage and computing time, which have to be solved in order to process the millions of data points, provided by the scanning systems. Also pre-processing systems do have specific requirements towards the surfaces, generated from the digitized data.

The aim of this project is to develop a software module, which provides the user with tools in order to solve the reconstruction problem from digitized data in a simple and effective way. The system to be developed supports the common scanning technologies and provides the user with tools to generate surfaces within their requirements. Due to the large number of applications and the simpler representation, the approach is restricted to the generation of triangular meshes. The module fits within the TEBIS CAD/CAM System, which is used in industry in many applications concerned with the design and manufacturing using computers.

1.2 Contributions

The contributions of this thesis are two-fold. First, the requirements and the preconditions for the surface reconstruction problem are derived. From this, a complete software solution for surface reconstruction, covering solutions to the major problems in this field, is derived and proposed. The solution covers

- Automatic reconstruction of triangular meshes from digitized data.
- Interactive functionality for optimisation and preparation of digitized data and triangular meshes.

Secondly, the most difficult task in this reverse engineering process, the automatic generation of triangular meshes from discrete 3D sample points is derived and presented. The approach is based on the principle of *multiple view combination*, which leads to a fast, reliable and storage efficient approach for the generation of such meshes. The method is an automatic process, which is incremental, behaves robustly and interpolates the sample points. Mesh optimisation routines for fairing and triangle decimation optimise the surface in terms of smoothness and reduce the number

of triangles for an efficient pre-processing, while preserving the shape and accuracy of the surface towards the data points.

Since the generated meshes are compatible to Stereolithographie (STL) file format, they can be further processed in all applications based on this representation.

1.3 Guide Through the Contents

The whole thesis consists of eight chapters and one appendix. The following paragraphs provide a chapter by chapter guide to the structure of the thesis.

In Chapter 1, the reasons for setting up this project are explained. The major contributions of this work are explicitly given. Also the original project proposal is presented.

The problems to be solved within this work are presented in Chapter 2. First the requirements to be met by a surface reconstruction system are derived. Since the TEBIS CAD/CAM System is strongly related to the automotive industry, this marketing analysis is done by examining different departments of companies such as BMW and DaimlerChrysler. As a result, a catalogue of requirements for a surface reconstruction system is presented. Then an overview over digitizing systems used currently in this industry is given and their strengths and weaknesses are compared. Finally, we identify and examine the problems, relating to this research.

A major result from the analysis in Chapter 2 is, that triangular meshes have to be generated from digitized data points. An overview on the literature, related to polygonisation and mesh optimisation is given in Chapter 3.

In Chapter 4, the necessary developments for processing sample data are described. A data structure for the representation of the different types of probed data is derived which is unique for all scanning systems. Then some fast and easy optimisation algorithms are presented. Moreover some examples are given and the results of the optimisation algorithms are discussed.

The data structures for the representation of triangular meshes are described in Chapter 5. A storage efficient structure is derived, which is used in this work. Also, a structure for the representation of a mesh at different levels of detail, the progressive mesh, is explained. Finally, different mesh manipulation operators are described.

In Chapter 6, an approach for the solution of the most difficult part in the reconstruction process, namely the automatic generation of triangular meshes from sample data, is presented. A fast, computer storage efficient and reliable algorithm for the automatic generation of meshes is presented. The algorithm is based on the concept of *multiple view combination*, which is discussed in the literature ([SL95a],[TL95],[Kar97]). Solutions are given to the problems of 2.5D triangulation, mesh limitation and gap-bridging.

Mesh smoothing and triangle decimation in order to optimise the reconstructed surfaces is presented in Chapter 7.

The results of the mesh generation process are presented in Chapter 8. Several examples for the reconstruction are given. Also, several examples for mesh smoothing and triangle decimation are given, as well.

Finally the conclusion is given and the major tasks for future research are discussed.

1.4 Original Project Proposal

In the process of car design and manufacturing there are many tasks, for which real parts have to be converted into computer models. With the development of optical and photogrammetric digitizing systems, real parts can be digitized very quickly and accurately. As a rule such systems produce digital point data (pointclouds). They describe the surface of the part very accurately and discretely with a predefined resolution. For further processing, these pointclouds have to be converted into triangular meshes or spline surface models using a postprocessing software system.

The surface reconstruction techniques used today can calculate surfaces, but the process is very time consuming. Especially in car-body design there are very high standards for the quality of spline surfaces and therefore the reconstruction process is very time consuming. Today it would be desirable to split up the surface reconstruction process into two steps:

- Generation of triangular meshes or tangential spline surface models with user defined tolerance and extensive automation.

- Surface optimisation based on meshes or spline surfaces to generate well structured, smooth and curvature continues high quality surface models.

One main advantage of splitting up this process into surface generation and surface optimisation is, that in an early state of the process a description of the surface already exists. These surfaces can be used for some of the next tasks, e.g. NC-Programming, Method-Planning, FEM Analysis etc. Because of the desired high automation, there is no special *Design* knowledge for the generation of the first surfaces required. Using the described methodology an important reduction of the development time of a product can be expected.

Therefore TEBIS plans to develop and produce a *Scan Module*. This software module mainly supports the following properties and requirements:

- Pointdata will be generated in a pre-processing step using mechanical, photogrammetric, optical or other digitizing systems.
- The next step will be data preparation using the *Scan Module*. Digitized data will be prepared and modified. Afterwards triangular meshes (STL-format) and spline surface models will be generated based on the data points.
- Using further preparation tools, the generated triangular or spline surfaces can be optimised, and are available for further processing, e.g. in the CAD/CAM functionality of TEBIS.
- Data points, triangular meshes and spline surfaces can be exported using standard interfaces such that further processing using other Software Systems can be done.

The aim of the software is therefore, to support the *state of the art* digitizing technologies and to reduce the amount of work involved in surface reconstruction. This will significantly reduce the development time in the design process.

Chapter 2

System design

Reverse engineering, in particular the generation of computer models from real parts, is required at many stages in the development of a product. In order to design a product, which perfectly meets the users requirements and which reduces the time and work involved in the reconstruction process, the industrial demanding have to be determined. It is necessary to derive, which tasks could be solved with the system and what applications should be included.

Also, in our special case the software to be designed depends on digitized data, which is provided from different digitizing systems available on the market today. It is important to derive the specific properties and problems of these systems. From these requirements, the functionality and algorithm to be developed can be derived.

In the following sections the requirements from the industry towards a surface reconstruction system are determined. Since the company Tebis, for which this research has been done, is strongly related to the automotive industry, the marketing research is done within the specific departments, namely the styling, design, pre-production, tool design, tool manufacturing and quality control departments, of this industry. Then the scanning systems are described and their specific properties and problems are derived. The analysis is restricted to digitizers, which are currently available on the market and used in industry. From this analysis, the requirement specification towards the software to be developed is derived and presented. The requirements from the marketing analysis as well as the requirements for the processing of digitized data from different scanning system are presented. Finally, the functional specification of the software to be developed is presented. This specification is the basis for further

development of algorithm for surface reconstruction from digitized data.

2.1 Industrial Marketing Survey

The design and manufacturing of the exterior and interior of a car, the *visible parts*, is done in different stages from the ideas of the stylist up to the manufacturing of the car. The different stages are illustrated in Figure (2). This design cycle, of course, is very abstract since usually the different stages do overlap and are not strictly sequential.

In any of these stages it is required to have real physical parts (real parts) and computer models (virtual models). Any department has different tasks and therefore different requirements towards a reconstruction system. The departments and their requirements for surface reconstruction are shortly described below.

- **Styling**

A *stylist* needs to get a good impression of the shape, of the functionality and the overall concept of a model. Most important for the stylists are real models. Still the *feeling* for the shape and the visual impression of the car is given best with an existing part. Also the classical *hand methods* of styling are much easier to apply since the stylist has direct influence on the part which he is manufacturing. He can *walk around* the model, touch it with his hands, handle it etc. Up to now this can not be simulated with software tools. Styling models are typically generated from clay within model making departments. These clay-models are easily modified by hand and therefore best suited for modelling. One important task in this styling process is to quickly generate copies from these clay models. If the stylist has a first model with which he is quite satisfied, he might want to try some changes in detail without destroying the original. Therefore he needs to duplicate his model twice or more times. Sometimes, only parts of the model are changed and an existing model should be copied with this changes integrated. This process has to be quick and accurate. Copy milling is one of the possibilities for the designer to generate a duplicate. The disadvantages here are obvious: It is a very time consuming process, only one copy at a time can be manufactured and the original has to be available at the time of copying.

Another important task for the stylist is visualisation. Latest software tools allow a virtual model to be placed into photorealistic surroundings. A car can be visualised

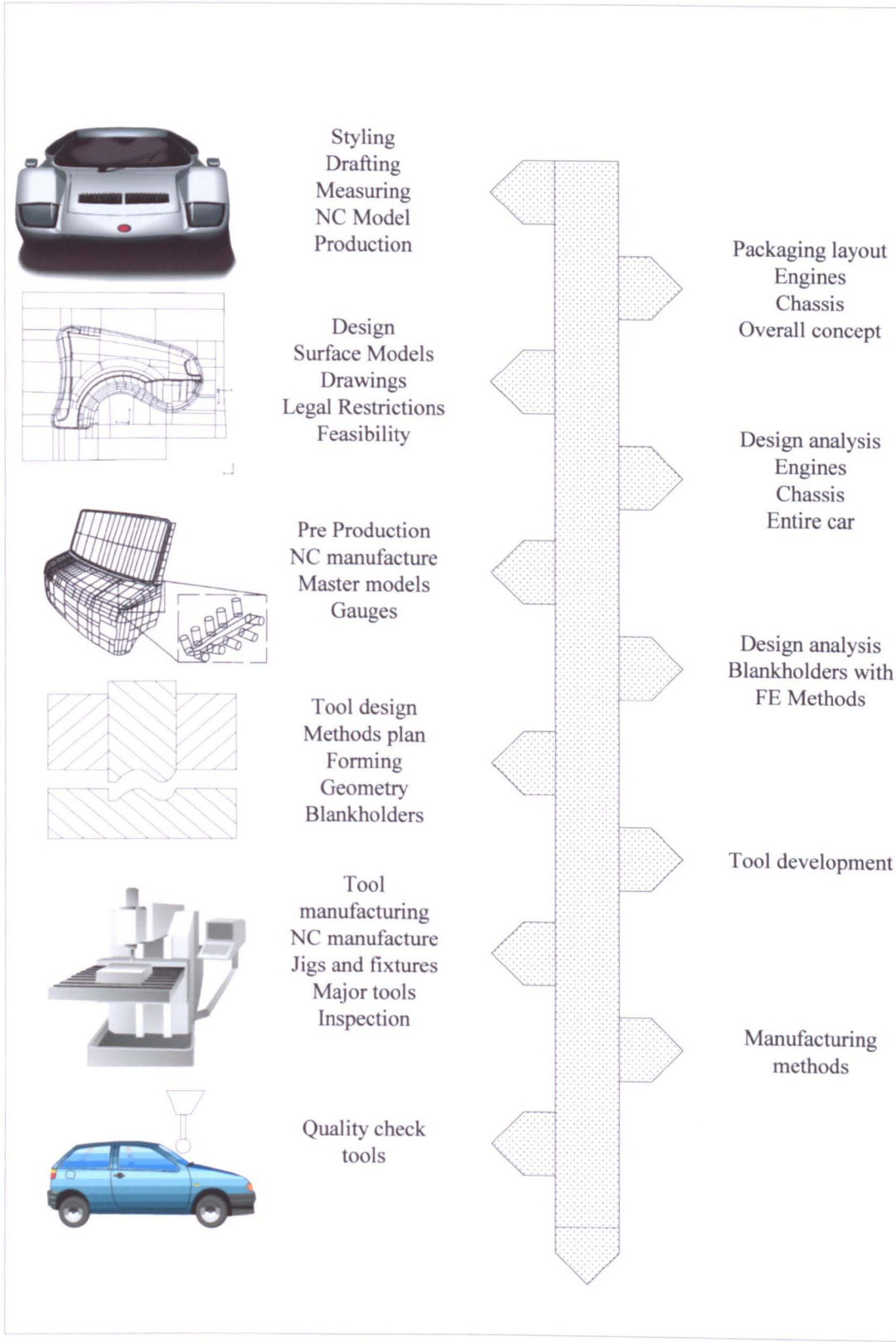


Figure 2: Design and Manufacturing

with natural colours and driven through virtual countries. With these tools the effect of the car in real world surroundings can be simulated. Also analysis tools such as finite element analysis can simulate the aerodynamics or deformation. For all these applications, the real model has to be converted into a computer model.

A clay model, which is accepted for further processing and manufacturing is called the *cubing*. It is the basis for pre-processing tasks.

- *Design*

The *designer* first creates a computer model based on the dimension of the cubing. Usually these design models are spline surface based models. The dimension of the model are obtained by digitizing. Since usually the CAD packages for surface modelling available today are not able to handle digitized data, the dimensions of the part are transferred via part sections into these system. The sections are generated within the digitizer software or special pre-processing software tools. At this early stage of the development, the surface models are used for analysis such as finite element analysis, legal restrictions, feasibility etc..

The *class A surfaces*, generated for the exterior of a car, need to have very high quality in terms of curvature, reflection and continuity. Even the continuity of the curvature is sometimes required for these surfaces in order to guarantee highest quality in the design cycle ([Hoc97], [HR98]). The quality of the shell of a car body is determined by projection of light stripes onto the parts surface. The reflection curve has to be smooth, continuous and pleasing. This reflection is more important for a stylist and designer than accuracy. So if a mathematical surface cannot be designed within a tight tolerance such that it has good reflection properties, then the tolerances are lowered.

Parts in the interior of a car must be more accurately measured since they have to fit into tighter dimensions. They also have to be nice and smooth but not with such a high quality as exterior parts. Of course the designer would like to generate surfaces with a quality such as Class A, but today's surface generation system does not allow them to be generated in a short time. Generating a Class A surface model of a car exterior is a very time consuming modelling process and today it easily takes a year or more to complete a model. Also the stylist changes his mind many times and recreates parts of his model. These changes have to be integrated into an existing model. All

these requirements on the surfaces have the only property: that from a computer model generated real part has a smooth, nice looking surface which also meets the designer's intention on the shape and which fits into the specified dimensions.

Many times in the design process, the stylist changes parts of the model which then has to be reintegrated into the current cubing and into the CAD model. Also within the design process, physical models have to be manufactured for further analysis such as feasibility or built-in analysis. Changes, which have been made due to these analyses, have to be reintegrated as well.

Finally, parts for the motor unit or gear box are not generated from clay models but constructed within software tools. Traditionally they are designed from two-dimensional drawings and consist of regular geometry rather than of free form surfaces. Only in recent years, the surface model e.g. of a motor unit consist of free form surfaces. Some of these surface are changed if they do not fit into the car or if problems arise due to blocking. Also, they are modified, if the components do not work properly in practice due to design errors or other physical problems. Some of the problems can be detected in advance with computer models and finite element tools. Some of them can only be detected in practical analysis. Most of the modifications to be done are applied manually during the pre-production process and have to be taken into account in subsequent stages.

- **Pre-Production**

From the models, generated during the design process, a *master model* is generated. The master model is a first prototype of the part to be manufactured. Feasibility tests, endurance analysis and other general functional tests are done with these prototypes. Also the manufacturing process can be simulated and checked with these models such that the later mounting of a car on assembly lines runs without problems.

Again, alterations due to changes in design, feasibility, built in problems or other more general problems have to be registered and taken into account for the manufacturing in later stages. Also gauges are constructed from the master model. They are used for quality checks in the later manufacturing process. Again the dimensions of the parts are the basis for constructing these gauges. Modifications have to be measured, reconstructed and reintroduced into the design models.

- **Tool Design and Manufacturing**

After the master model has been generated, the tools for the manufacturing of the car are produced. A variety of different tools have to be manufactured in order to produce the different component parts.

Compression moulding dies are manufactured, from which the sheet metals for the shell of a body are made. The sheet metals are deformed with these tools into the desired shape. Injection moulds have to be fabricated in order to manufacture the synthetic parts (plastics).

These tools are manufactured almost exclusively from virtual models, generated from the design department and with software tools which control the manufacturing machines. Today's software modules guarantee a fast and accurate generation of these tools. Although they are accurately manufactured, the toolmaker often has to change the model manually. Sheet metals usually get distorted after the deformation. Only with manual changes, which are applied to the tool with the special knowledge of the toolmaker, the sheet metal will get the desired shape in the deformation process.

Computer simulation programs can reduce the amount of time, involved in the manual fabrication of tools. There are software tools available, which can simulate the flow behaviour of a sheet metal or of the granule in the injection moulding of duroplastic material. Tools for calculating the removement of a part from a tool simulate the behaviour of the part after deformation. The calculated models, created from finite element based simulations, can be compared with the original models and problems in the manufacturing can be corrected in advance. Despite of these virtual simulations, a manual correction of the tool cannot be avoided up to now.

At this stage of the design cycle, the master model exists only in terms of tools, from which the model is manufactured. The manufactured parts can be different in terms of their dimension to the master model, generated in the pre-production process, if the stylist is still satisfied and the changes are necessary in order to guarantee an error free assembly. Since many of the very expensive tools only exists once or twice, the dimension of these tools have to be registered, documented and archived. Some tools wear out during the manufacturing and assembly process. Therefore they have to be controlled from time to time since major changes can result in problems during assembly. These changes might only be small and do not effect the dimensions of the generated parts, but in worst case they do make the tool defective and useless. Reproducing the tool is very time consuming and cost intensive, if it must be done

with the same amount of manual work as for the first one. In order to generate duplicates of the tools quickly and cost effectively, their dimensions are determined in advance and the master model is reconstructed within a CAD software. Again this can be done by digitizing the surface of the part and by surface reconstruction. Also during the manufacturing process, the tool is digitized from time to time in order to measure the aberration. The comparison of the measured data with data from the master form gives an overview on the amplitude of aberration. Changes can then be detected before problems arise, which might lead to an interruption of the assembly.

- **Quality control**

Finally the tools and generated components have to be checked wether they meet given tolerances, fit together and meet the stylists intention. The accuracy of a single part is very important since all parts have to fit together into a common model and an error free assembly has to be guaranteed. In a process such as the manufacturing of a car, any part must be accurate and problems have to be detected very early.

One of the tasks of quality control is to control dimensions of the tools and parts, which are involved in the assembly process. Many of the parts can be checked with the gauges, manufactured in the pre-production stage. Some of the parts and the tools have to be measured and compared with the master model. Again, problems may arise due to aberration, attrition or equivalent. It is important to detect these problems in advance in order to correct them before problems lead to an interruption of the assembly. Digitizing the parts and tools are an opportunity to detect and control changes.

2.2 Marketing Specification

There are several application for which digitization and surface reconstruction is required. The following applications should be solvable within a surface reconstruction software.

Of course, the reconstruction software should support the digitizer used in the industry and in particular the specific types of data, produced by these systems. An analysis of the digitizer is given in Chapter 2 Section 3. A surface reconstruction

system should be able to incorporate native file formats as well as it should support standard interfaces such as VDAFS, IGES or STEP.

The first task to be solved with digitized data is variance comparison. The CAD system, for which the software will be developed is able to process polynomial surfaces. The distance of the digitized data points from the spline surfaces should be measurable and the result should be documentable. A good visualisation of variances can be achieved with triangular meshes. Coloured shading can be applied to these meshes, and therefore the variances can be visualised more distinctively.

Visualisation is also required for virtual reality applications. There are standard file formats such as STL or VRML[1] for the data transfer to such systems. Since animation within these systems should be fast, the number of triangles should be minimal while the shape should be accurate. For applications such as the presentation of products in internet shopping centres, the accuracy is not so important but the number of triangles heavily influences the speed of computation and therefore should be very low while the shape is still recognisable. There should be no holes in the mesh and no degenerate[2] triangles since these problems produce artefacts within the visualisation application. Also the surface should be smooth and pleasing.

The traditional application for STL files is the stereolithographie itself. The triangular meshes to be produced should be compatible with these meshes. Therefore they have to be topologically and geometrically closed (*watertight*). This means that any corner of a triangle (*the vertices*) must only be adjacent to other corners and not to edges. Also any halfedge of a triangle must be adjacent to exactly one other halfedge to guarantee topological closeness. This is explained in more detail in chapter 4.1. Also some stereolithographie systems are not able to process self-intersecting meshes. Those geometrical problems must therefore be avoided.

Some CAD software packages are not able to process digitized data or triangular meshes. Traditionally a CAD package is based on polynomial curves and surfaces such as B-splines or NURBS. These surfaces are continuous and differentiable so that surface normals or curvatures can be calculated. It is also a very storage efficient representation. A common way of transfering the geometry of digitized models into

[1] Virtual reality modelling language

[2] Degeneracy can be of different kinds. Single outliers or holes are unwanted artefacts. Also small triangles produce artefacts since the calculation of the normal vector, which is required for shading and reflection, is unstable.

these systems is to generate plane part cuts which result into polygons or spline curves. These curves can then be exported via a standard interface such as VDA or IGES and imported within any CAD system. The part cuts should be accurate, optimised in terms of smoothness and they should consist of a low number of segments since the amount of data to be transferred must be small and pre-processing systems are limited in terms of storage.

CAM packages and in particular software systems for the generation of milling toolpathes are also able to process triangular meshes. There are two different applications for the processing of meshes, generated from digitized data.

The first application is the duplication of a digitized model. A model can be duplicated analogus to copy milling by generating toolpathes for manufacturing the shape on milling machines. Here one copy at a time is produced. Another way is to manufacture injection moulding dies, compression moulding dies or other tools for the manufacturing of more than one duplicate. In this case, the complete die has to be modelled either with meshes, with spline surfaces or with a mixture of both representations. These mixed models are called *hybrid models*. The advantage is, that complex shapes can quickly be obtained from digitizing and mesh generation. Interactive spline surface generation then only has to be done for the completion of the tool. Still there is functionality required for the preparation and optimisation of digitized data and for the meshes. Of course the meshes should be smooth and pleasing such that the manufactured surface is also smooth and no further manual finishing is required. Also the amount of storage required should be minimal for fast and efficient computation. For the design of a tool there are more tasks to be solved. Useless data such as data from the measuring table must be removable and holes should be closable. The reconstruction system must be able to split up a closed model into two halfs for the generation of the upper and lower die. There also should be functionality for the generation of smooth blending between meshes and spline surfaces. To conclude, the digitized object and in particular the mesh surface should be handled in the same manner as spline surfaces are handled in CAD modelling applications. This guarantees intuitive and easy handling for a CAD experienced designer.

Roughing is the second milling application, where digitizing and surface reconstruction can optimise the process. A roughing tool removes the material from a

blank and generates a rough shape of the model, which can then be finished. The blanks are usually mould castings. The geometry of the blank is often not known exactly in advance. This geometry can be obtained with reverse engineering and taken into account in the roughing process. If the blank is known in advance, the toolpathes can be optimised such that the milling tool always removes material. The geometry of the blank is describable by triangular meshes. More information on this technology can be found in the work of Schuster [Sch00].

Triangular meshes can be processed using the finite element method (FEM). The requirements for the quality of the meshes are very high for these applications. Today's FEM systems can only handle a certain number of triangles. The solvers are only able to calculate models up to 100000 triangles and still require days of computing time on high-end computers. Also the systems are very sensitive towards the triangle geometry. The *aspect ratio*[3] of a triangle should not be less than a certain value. The better the ratio is, the better and more accurate is the calculation. The model should also be accurate even in areas of high curvature which is a requirement, which has not been solved up to now.

Finally there are software packages for solid modelling based on *mock-up solids*. Mock-up solids are defined by piecewise planar surfaces, which are described by planar boundary polygons. Triangular meshes can be formulated as a subset of these solids. If this definition is given, the meshes are transformable into this representation.

Summarising the following tasks are given from the marketing analysis towards digitizing and surface reconstruction :

- Supporting the common, available digitizer.
- Variance comparison of digitized models towards given computer models.
- Generation of smooth and pleasing surfaces without artefacts, compatible to visualisation software tools.
- Generation of VRML compatible surfaces with low storage requirements for the display and animation in virtual reality software tools.
- Generation of geometrically and topologically closed, non self intersecting triangular meshes, compatible to the stereolithographie file format.

[3]Aspect ratio is the ratio of the height of a triangle towards the longest edge.

- Calculation of plane part cuts through digitized objects for the transfer and pre-processing of these curves in standard CAD packages.
- Generation of meshes or surfaces for copy milling applications.
- Design and modelling of mould, die and pattern tools from reconstructed objects.
- Generation of closed solids for the processing in roughing applications.
- Generation of high quality triangular meshes, compatible to finite element analysis tools. High quality in terms of triangle roundness, closeness and accurateness while the number of triangles should be minimal.
- Mock-up solid generation from digitized data.

2.3 Data Acquisition

There are several different technologies for digitizing real models. All these methods produce a discrete set of 3D data points. These data points are either unstructured, structured in 2D matrices or ordered in scanlines. Each method has strengths and weaknesses which will be discussed in this section. First, the technology of each scanning system is described. Then the problems, which arise due to scanner limitations are worked out and discussed. An overview and further literature on scanning technologies is given in the survey of T.Varady et al. [VMC97].

- **Tactile measuring**

The most common method for digitizing is the mechanical (tactile) method. Tactile systems have been used to digitize models for several years. There are numerous different systems available on the market.

A sensing device, which is located in the joints of a mechanical arm, touches the surface. The arm is part of a co-ordinate measuring machine (CMM). The sensing device is usually a sphere which touches the surface, in combination with some release mechanism. The generated data point is equivalent to the midpoint of the probing sphere. Figure 3 shows an example of such a sensing device.

Figure 3: Tactile measuring system

The CMM can be programmed, such that the sensing device follows specified paths along the surface and automatically follows the shape of the part. Data points are then generated automatically from the machine control device.

There are at least three different strategies used in practice in order to digitize a part. The most common strategy is the ax-parallel, bi-directional. The tool is moved parallel to one axis (y-axis) of the CMM bi-directionally along the part. Sometimes, an axis-parallel one-way strategy is used. This has the advantage, that specific properties of the machine are reflected constantly in every scan line. The scan lines are more *harmonic*. Nevertheless, this method is more time-consuming. A less common used strategy is the *circular* method. The tool is moved in some sort of *snake* form along the part. It has the advantage, that the part is digitized at any time of movement.

A simple strategy used in practice and recommended for digitizing a part is first to generate a main program for the whole part with a one-way or bi-directional strategy. Scanlines are produce with a constant line feed motion such that the resulting scanlines do have a constant distance to each other in the scan plane. The 3D distance of the scanlines to each other depends on the steepness of the part in the direction of line feed. Surfaces with a large angle to the scanning direction lead to large distances between the scanlines and therefore do not describe the surface with a sufficient resolution. Therefore secondary programs are produced by changing the direction of

movement of the tool, such that all steep surfaces, sharp corner or small radii are digitized with the tool moving crosswise to the corner. The combination of these different programs (also called *scan records*) describe the part sufficiently in most cases and give a good basis for surface reconstruction. In practice a part is often digitized with two, crosswise over the whole part generated scan records, since this can be done automatically without user interaction.

More expensive and therefore less common methods are optical scanner such as photogrammetric and laser scanner. In contrast to the tactile methods they are non-contact methods.

- **Photogrammetrie**

Photogrammetrie is the capturing and analysis of photographical images. The evaluation of these images is done with methods of digital image processing. One commonly used method of non contact digitizing with photogrammetrie is based on this technology. Structured lighting is projected onto an objects surface in a fixed scan direction and is recorded with two integrated video camera systems. 3D data points are generated using geometric triangulation and image processing methods. Figure 4 illustrates the principle of structured lighting.

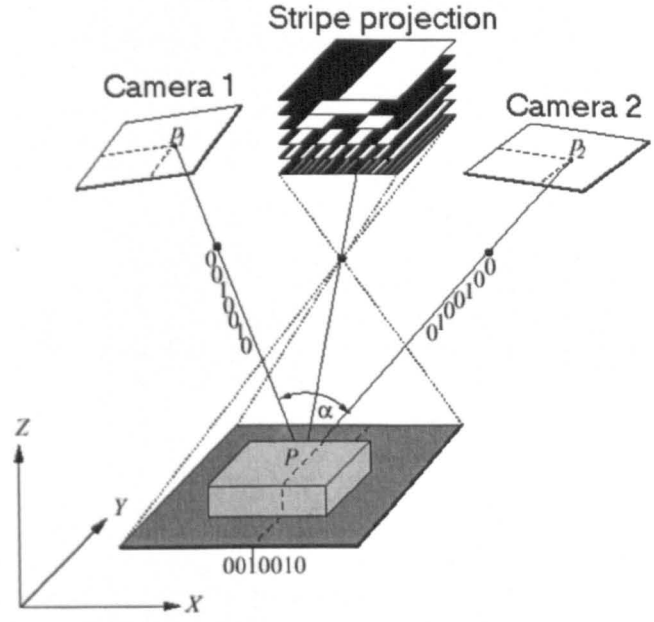


Figure 4: Principle of structured lighting

The light source and the cameras are mounted onto a travelling platform with which multiple scans of the surface are produced as shown in Figure 5. Structured lighting systems are available with different photosensitive devices in different quality and resolution. The biggest systems have a field of view of a square meter, a resolution of 1280x1024 pixel which to lead up to 1.300.000 data points per view and an accuracy of down to 0.2 mm. Smaller systems (down to 50 square mm or with lower resolution) have a better accuracy, but the field of view is smaller.

Measuring and alignment of multiple scans is also done with photogrammetric methods. First, specific basing marks are brought up to the object. Then the object is captured from different views using a digital camera such that at least three basing marks do overlap in adjacent views. The 3D co-ordinates of the centre of these marks are calculated from the different overlapping basing marks in each image. Finally, when digitizing the object with structured lighting, the single range images are transformed into the global co-ordinate system using the 3D co-ordinates of the basing marks. The marks leave holes in the scan records, produced with structured lighting. For alignment these holes have to overlap the basing marks, which are already transformed into a common co-ordinate system.

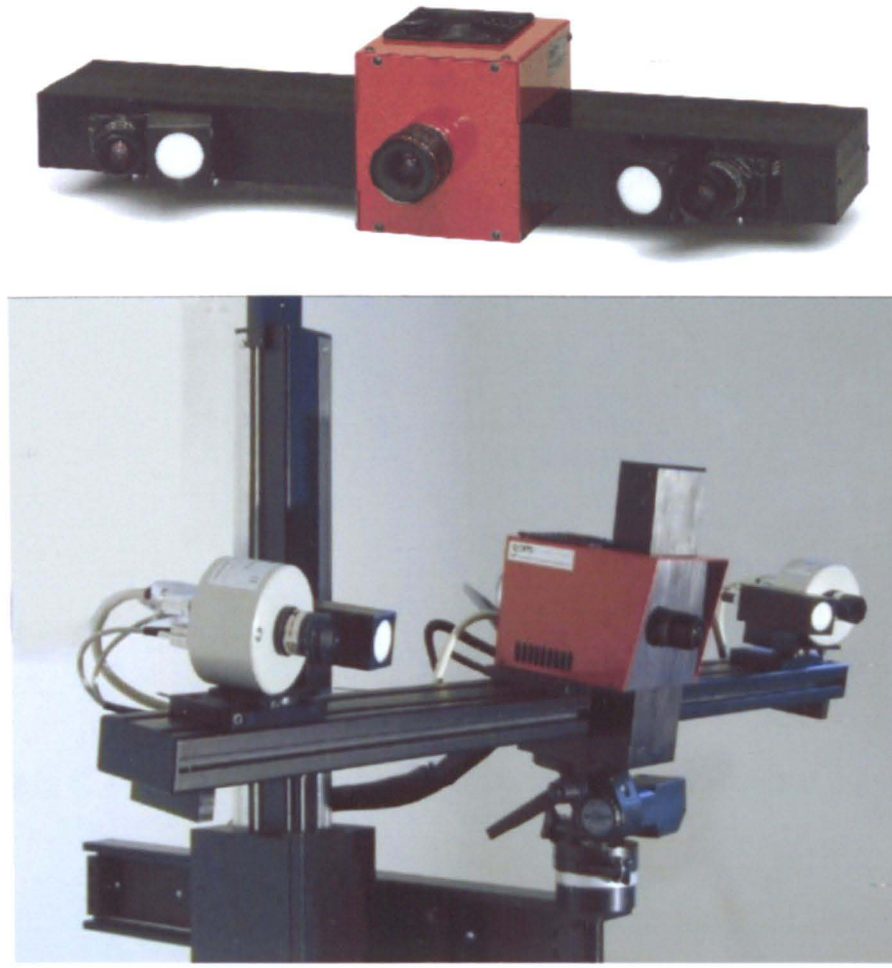


Figure 5: Structured lighting sensor.

- **Laser scanner**

The most common laser scanner is based on the method of active triangulation. A laser beam is projected onto a double-sided scanning mirror which provides synchronisation between projection and detection. Data points are calculated by analysing the *time of flight* to determine the distance travelled. Figure (6) shows the principle of a typical range laser.

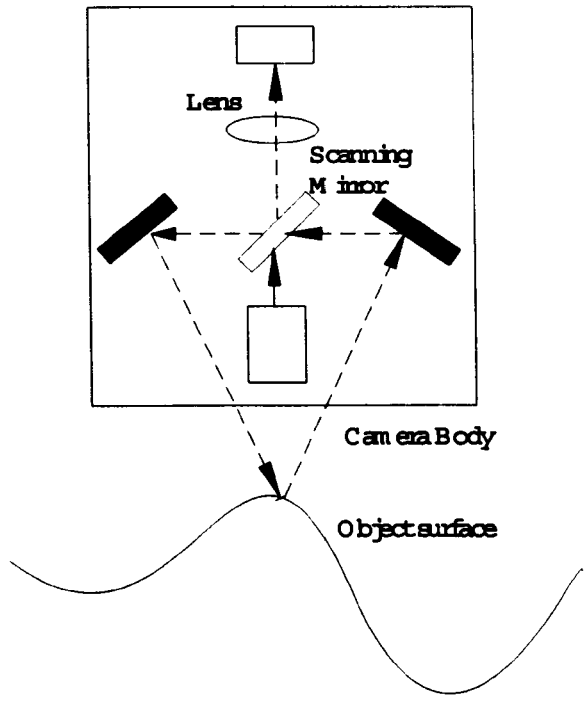


Figure 6: Principle of active triangulation using synchronised scanning.

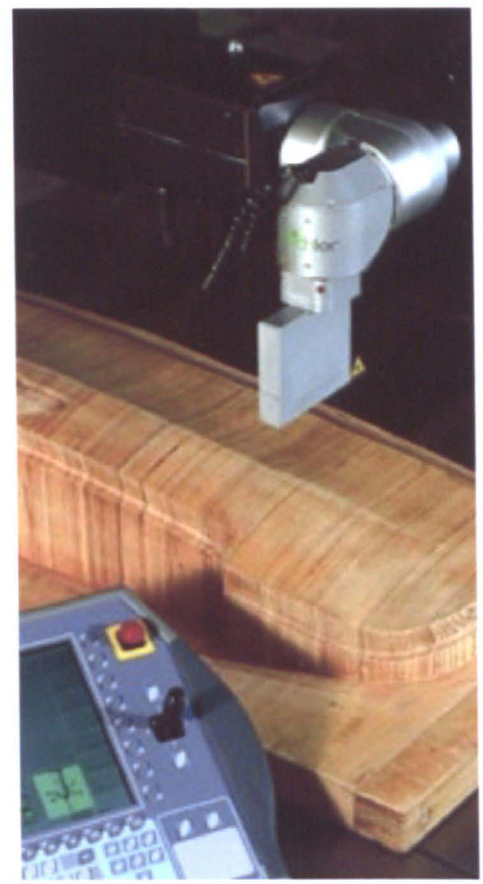


Figure 7: Laser mounted on a Co-ordinate Measuring Machine (CMM).

A laser scanning device is typically mounted on a translation device such as a CMM, and data is acquired with the camera in motion (Figure 7). A beam of laser light is emitted and usually swept across the scene through a predefined angle. If the camera is stationary, all acquired data will lie in a plane. The field of view is a region within the plane for which surface sample data is generated (Figure 8). Laser systems are able to produce data points with a very high accuracy. The field of view is limited by the CMM limitations and the area of obstruction.

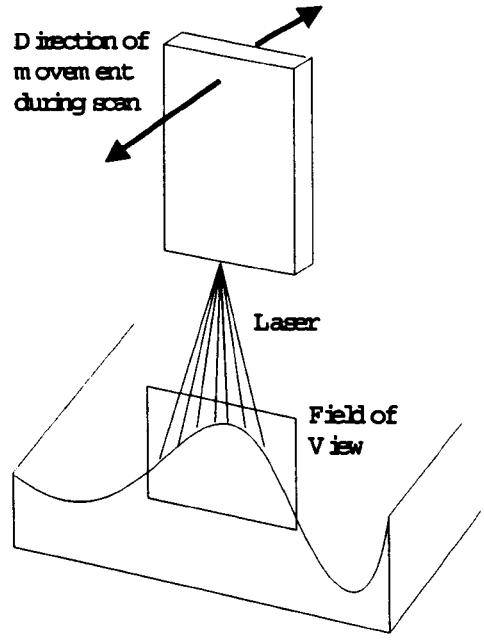


Figure 8: Digitizer field of view.

The density of the pointset in the direction of movement depends on the speed of the CMM. The laser beam while being swept across the part generates a fixed point resolution (e.g. one point per 0.1 mm) within the scan line. Laser systems generate a very high number of data points in short time compared to tactile systems. There are also range laser systems available, which are mounted on a measuring arm with 6 or more degrees of freedom. An object is measured by the user, moving the laser analogue to a *paint-brush* along the part. This system is, in contrast to all other methods, extremely flexible since the laser can always be placed and oriented optimally to the surface.

Some laser scanning systems combine more than one scanner to capture a closed object such as a human body with one scan at a time. These *body scanners* are able to capture a large object from different views within a few seconds.

Laser alignment is done in advance to the scanning process, usually in combination with the CMM. This involves determining the orientation of the scanner with respect to the CMM arm, as well as the positional offsets of the arm to a reference object such as a sphere or plane. If the laser orientation is changed in order to capture the object from different views, the laser has to be realigned. There are specific systems available, where this is not necessary and the orientation is compensated in

a pre-processing software, knowing the transformation co-ordinates for each degree of freedom. After scanner alignment, all scan records are acquired in a global co-ordinate system. Alignment of multiple views within the pre-processing software is therefore not required.

- **Comparison**

Tactile systems are very robust and they produce accurate data. Because of the sphere tool, the system is less sensitive on surfaces with a bad surface finish. Tactile measuring is very slow and therefore not useful for digitizing large models, such as complete cars shells. Also, soft or deformable surfaces can either be damaged or the result is not accurate. Different forms of the CMM arm and sensing device allow the detection in difficult areas such as through holes or areas of occlusion. Occlusion is the blocking of the scanning medium due to shadowing or obstruction.

Typical problems of tactile measuring with bi-directional movement of the digitizing sphere are shown in Figure 9 and 10. Figure 9 shows the effect of moving a CMM to fast over the surface of a part. In Figure 10, the scanlines overlap each other due to different ways of movement for each direction. On the left side an example is shown, where neighbouring scanlines differ and wrongly describe the corner. In the example on the right side the neighbouring scanlines do overlap each other, even if they should have a constant line distance towards each other. Both problems arise with bi-directional scanning.

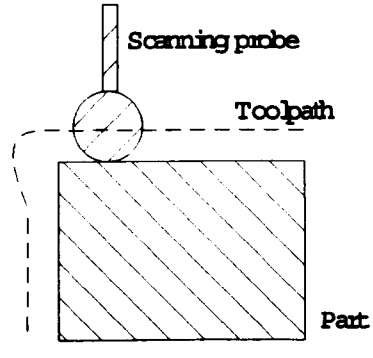


Figure 9: Problems in data acquisition with tactile measuring due to the speed of the CMM.

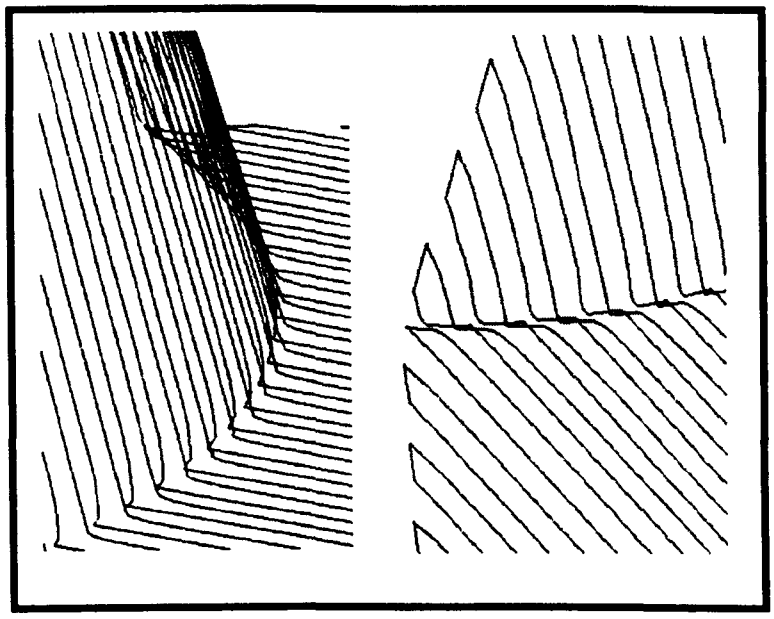


Figure 10: A typical problem in tactile measuring with a bi-directional scanning strategy.

Since every data point is not a point on the surface but the midpoint of the sphere tool, the real surface has to be recalculated by some radius compensation algorithm. Even so not every corner or small radius can be recalculated. A convex corner in the part will lead to a corner in the toolpath. Reoffsetting the corner usually leads to a radius in the resulting surface. The problem of scanning convex corner with sphere tools is illustrated in Figure 11. Figure 12 shows an example of a toolpath with offset.

Features such as convex corners have either to be redigitized using tools with smaller radii, or they have to be redesigned in the surface reconstruction system if pre-knowledge of the feature dimension is given.

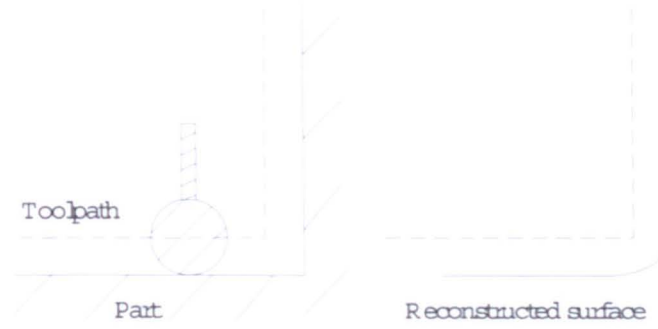


Figure 11: Problems in digitizing convex corner with tactile measuring and sphere tools.

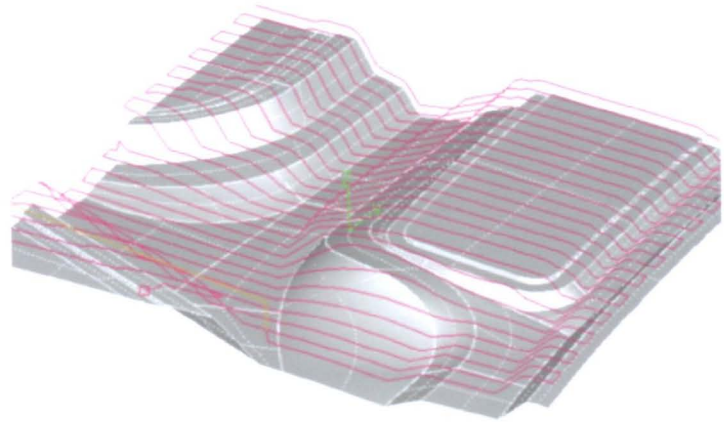


Figure 12: Data with offset, produced by a tactile measuring system.

Photogrammetric systems are more flexible. The system can be moved to the object to be scanned and the object need not be fixed. Objects can be digitized very quickly since it takes just a few seconds to generate a high number of data points (except installation and calibration). The resolution and accuracy of the generated pointset depends on the resolution of the different video optics. Also there is no offset compensation necessary, since the data points are directly on the surface. With a photogrammetric scanner, a model can be captured very quickly with good accuracy. In practice this accuracy is good enough for many applications.

The surface finish has a great effect on the quality of the data. The system has problems with dark or shiny surfaces and is only able to capture visible areas. Steep areas with an angle of more than 60-80 degree to the scanner orientation cannot be digitized. Because of a fixed resolution it is not possible to generate a denser pointset

in areas of high curvature, at sharp corners or at surface boundaries. In practice, these features are measured afterwards with a tactile system, with an optic with higher resolution or they have to be reconstructed within the pre-processing software.

A great problem in photogrammetrie is alignment. There are two different strategies in order to register measurements from different views. One approach, mainly used in industry, is to apply basing marks to the object. These marks are measured in advance with a tactile system. They are then used to register the single views into a global co-ordinate system. This method is accurate but slow and an additional measuring system is required. In the second approach, measurements from different views are registered using alignment (so called *matching*) operations in a pre-processing software. The different views are aligned towards each other using a distance minimisation approach. This method has several disadvantages and fails if the object has no characteristical features. Two scan records from the roof of the car are almost plane and the alignment is ambiguous. Using a matching algorithm it is possible, that an object shrinks or enlarges which is not desirable. Figure 13 shows an example of two scan records, which do not match due to alignment problems.

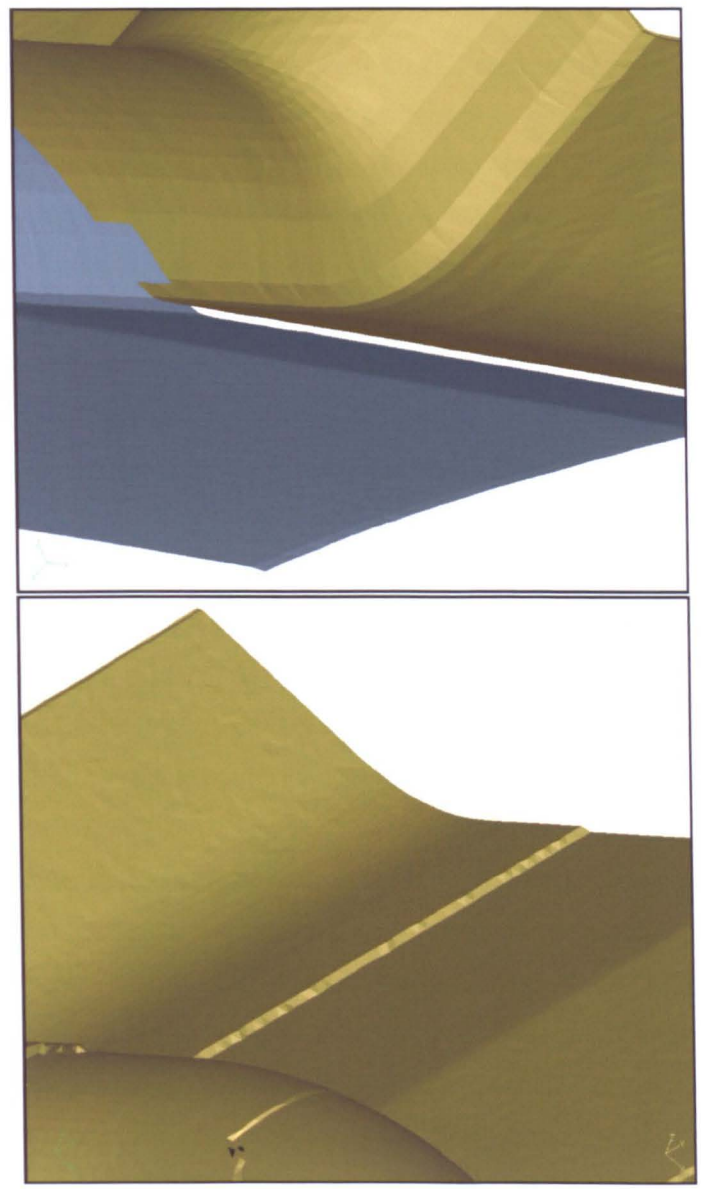


Figure 13: Mismatch between different scan records due to calibration problems

Finally, digitized parts of an object such as a car have to be transferred into a global *car co-ordinate system*. Usually there are basing marks fixed with every object, from which this transformation can be calculated. These basing marks also have to be measured very accurately and taken into account in the calculation of the scan record transformation.

A Laser scanner has to be moved along an object with an additional CMM. The measuring machines are either stationary or flexible. Stationary systems are less

flexible but more accurate; the average error to the real surface of a high quality laser scanner lies between 0.03 to 0.1 mm. This accuracy is good enough for almost every application in car industry. Combined with an intelligent numerical control, the laser can produce very quickly data for a complete model, with dense data in areas of high curvature and less dense in plane areas. An intelligent machine control also enables the laser to follow along specific paths on the part such as tapes or sharp edges and produce optimal scanlines crosswise to the corner, from which the feature curves can be recalculated. Laser scanners can also be placed on different CMMs for different applications. The more flexible the CMM is, the better results can be produced in difficult areas. Calibration has to be done once before scanning and the object can be digitized from different views without recalibration.

All methods are limited by the complexity of the part, optical more than tactile methods. Through holes and hidden areas cannot be detected with optical systems. The smallest radius to be detected depends on the touching sphere or the resolution of the optical systems. Any system has problems measuring the surface in areas of occlusion.

The problem of occlusion might be solved in the future by computer tomographic systems (CT-scanner). Data points are generated by analysing the density of the material of an object. Today CT-scanners are almost exclusively used in medical applications. They generate some sort of *plane* cut through the part at the boundary of different material densities, which can then be used for analysis or reconstruction. In the car industry, parts such as the engine or gearbox might be digitizable with these system.

In practice, tactile methods are used in areas, where accuracy is more important than time. Since tactile measuring machines are cheap, they are affordable and therefore widely used. Objects are digitized automatically, often overnight. Many tool and mould design departments use tactile measuring machines for reproducing their tools. Laser systems are faster and accurate enough to compete with the tactile systems, but they are more expensive and therefore less common. Photogrammetric system are used in areas where flexibility and speed are more important than accuracy. This is given in the styling departments of the car industry, where the scanner has to be moved to the object rather than the object to the scanner.

Figure 14 shows an example for data, generated with a tactile measuring device. The object was digitized from one direction of view, but with four different directions of movement. This was necessary in order to sufficiently digitize the dome in the middle of the object. Note, that the different range views only overlap slightly.

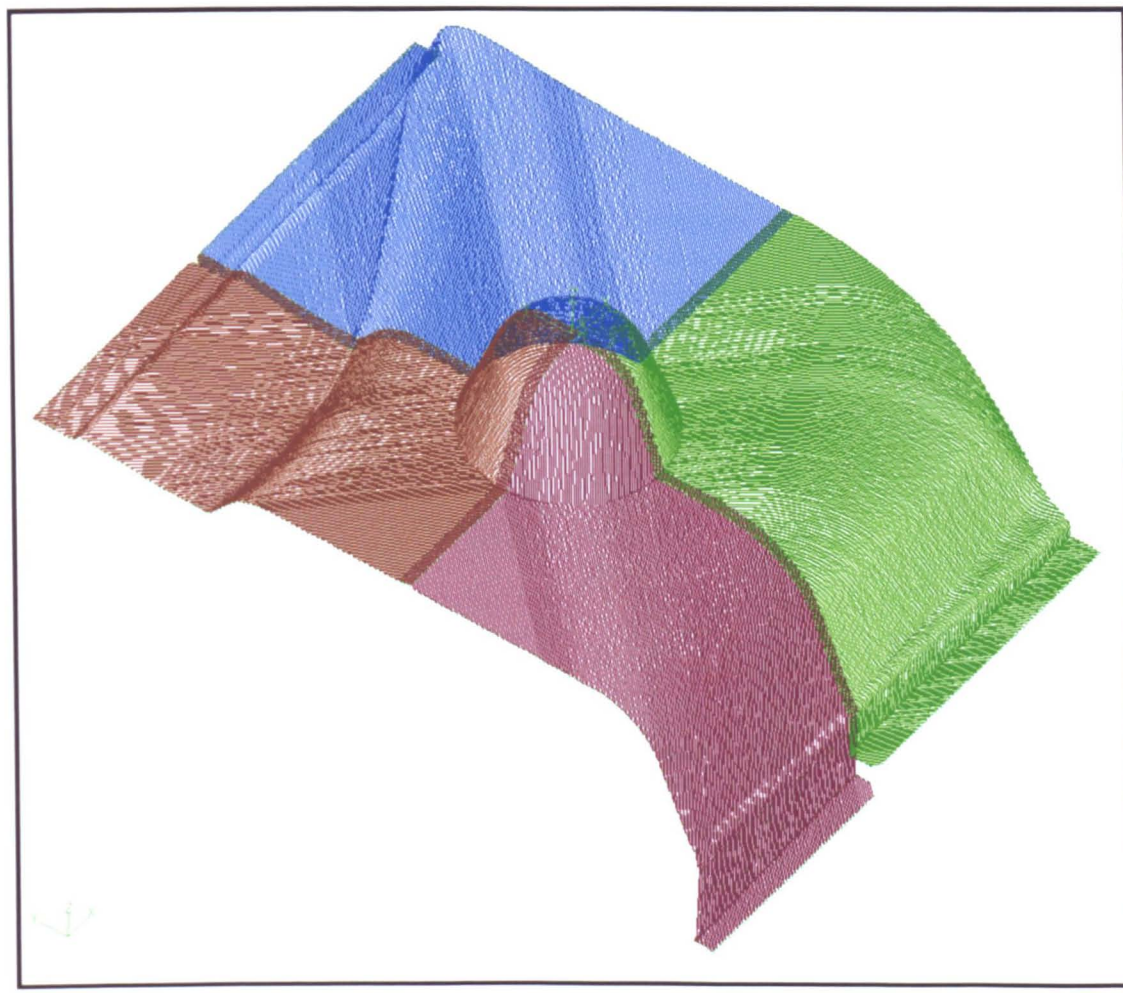


Figure 14: Example of digitized data from tactile measuring

Figure 15 is an example of an object, digitized with structured lighting. The object was captured from 7 different points of view. There are holes in the object due to the basing marks and due to problems with reflection. The range views overlap much more than in the tactile example.

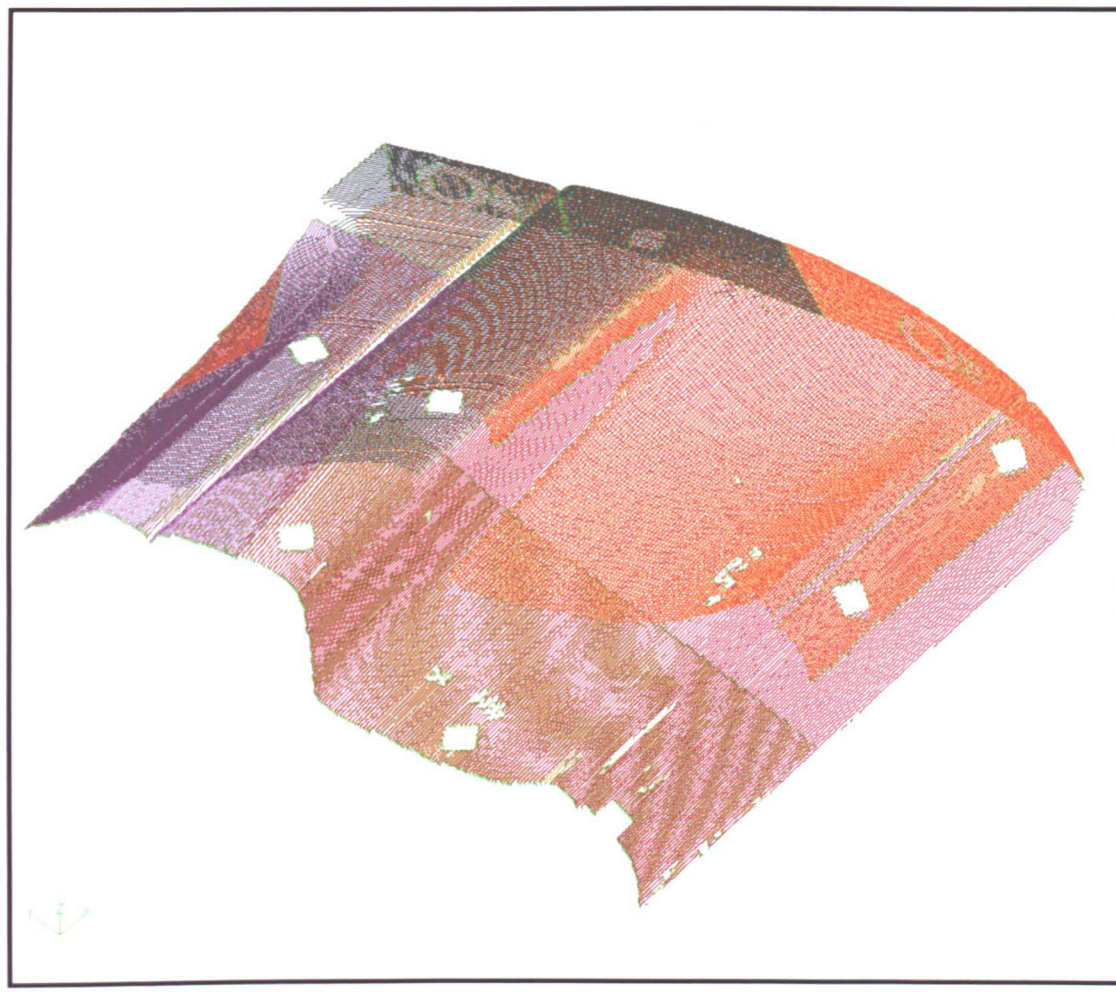


Figure 15: Object, captured with photogrammetry system.
Notice the areas, where data could not be obtained due to the basing marks.

An example for laser scanned data is given in Figure 16. The object was captured from only one point of view, but with 9 different directions of movement. Again, the range views do overlap heavily. Also notice the structure of the scanlines. They are almost parallel, but the distance between two scanlines varies. This is because the laser was moved manually over the object rather than numerically controlled. Also notice that there are areas, where the object could not be digitized due to obstruction.

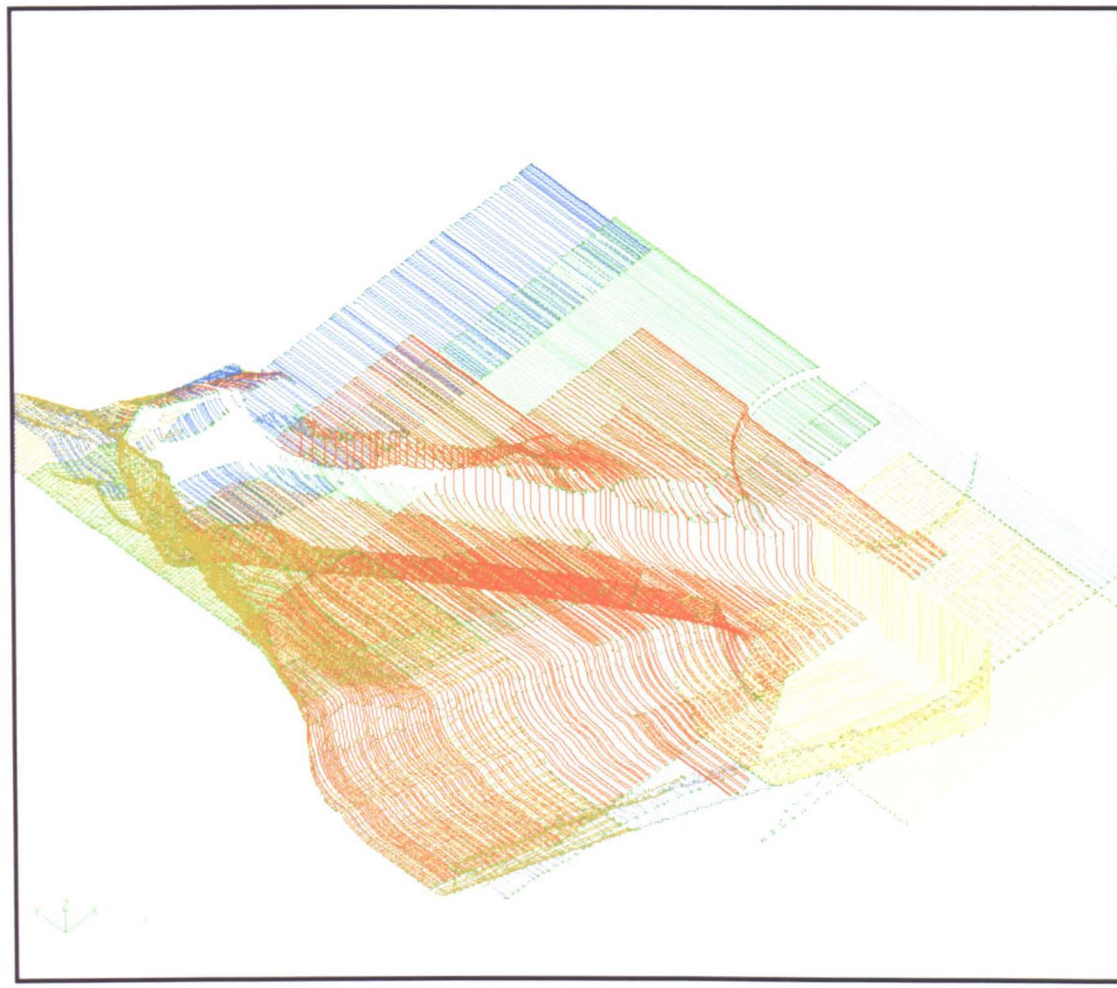


Figure 16: Object, measured with a range laser system.

2.4 Requirements Due to Data Acquisition

As described in section 2.2 one aim of this work is to process digitized data, provided from the different digitizers, available on the market today. All these scanning systems have different strengths and weaknesses. A surface reconstruction should provide the user with functionality in order to solve the problems, resulting from the scanner limitations.

First, the software to be developed should be able to process different types of digitized data provided by the scanning systems. Furthermore it should be compatible enough to support systems to be developed in the future. Since all digitizers produce

3D scattered data points, the system should be able to process these *point clouds*. Still there is more information given by digitized data, which might be helpful in the reconstruction process. To support this additional information, the system should incorporate the data via native interfaces.

The system should be able to handle different types and qualities of digitized data. Noisy data should be smoothed. Typical problems such as single outliers (spikes) in the data must be ignored or removable. Since some of today's scanning systems generate many millions of data points, the system should be able to handle or reduce these large numbers of data and hence the required storage, without the loss of information and accuracy. It should also be possible to delete data in areas of no interest. The measuring table or the fixturing is not relevant to the object description and therefore these features must be removed from the model.

A special problem arises from data, provided by tactile measuring machines. The scanlines are *offset polygons*, since the generated data points are not on the surface of the part, but the midpoint of the sphere scanning probe, when touching the surface. This offset has to be compensated in order to describe the original surface rather than an offset surface.

Not all areas of an object might be digitizable. This depends more or less on the scanning system and is a result from limited accessibility or occlusion. For an application such as stereolithographie it is required to generate a close object even if digitized data points are not available. Therefore functionality should be available in order to complete and close the model. This has to be done for simple holes such as those, remaining from the basing marks in photogrammetry data. But also more difficult geometry might be missing in the data and the user should be able to design and model surfaces for these areas and incorporate them into the model. These surfaces must be triangular meshes for applications such as stereolithographie or finite element analysis. They also could be polynomial surfaces. The resulting *hybrid* models, consisting of triangular meshes and spline surfaces can be processed in milling, viewing and design applications.

In order to save time in the digitizing process, often only one half of symmetrical objects are digitized. The reconstruction system should be able to mirror the digitized data in order to generate the whole object. The two sides should then be connectable and a smooth continuity should be generatable.

Before an object can be digitized, every scanning system has to be calibrated. This is usually done by the scanning software and it would not make sense for a pre-processing software, to develop calibration methods for all different scanners. Unfortunately, there are sometimes calibration problems within the scanning system which lead to steps or uniform mismatches in multiple views. Since these errors might only be detectable during pre-processing, a function to minimise this error is sometimes required. Also, sometimes only small modifications of a model have to be made. Calibration and alignment of the scanner and most importantly the alignment for digitizing the area in an object co-ordinate system is a very time consuming process in comparison to the scanning time, which only takes seconds. If the original CAD model of the object is given it would be desirable to align the digitized area towards the CAD model within the software.

Very important is the processing of multiple views. Since all scanning systems digitize an object from different scanning views, the software should be able to combine these views in order to remove redundant data and generate one surface describing the object.

Finally, important parts such as feature curves or small radii cannot be digitized using most of the scanning systems. In some applications such as styling, these feature curves are the most important parts of an object and should therefore be reconstructable and represented in the computer model. The features are either digitized separately with special systems, or they have to be reconstructed from the data available. Reconstructing these features would save digitizing time and money for special digitizing systems. These features have also to be incorporated into the computer model either during generation or in subsequent preparation steps.

Summarising, the following preconditions are given due to the limitations of the scanning systems:

- Data from different scanning system and multiple views.
- Processing large numbers of data points.
- Noise, unprecise data and single outliers.
- Redundant data in areas of no interest.
- Missing data due to accessibility, occlusion or alignment marks.

- Offset scanlines due to the radius of a sphere probe.
- Symmetrical objects, digitized only one-sided.
- Discontinuities due to calibration problems.
- Modifications to be aligned into the objects co-ordinate system.
- Insufficient digitized features such as sharp corners.

2.5 Functional Specification

After determining the requirements needed for a surface reconstruction system, the functional specification can be derived. Since the software to be developed fits into the Tebis CAD/CAM System, there are certain important pre-conditions resulting from the given functionality in the package. These preconditions influence the design of the module since functionality is given and the package must not be developed as a stand alone module.

First, there is functionality available in the CAD package to generate spline curves, surface, solids etc. The functionality is complete and sufficient to design complex computer models or tools. Secondly, the CAM module can be extended such that the milling functionality is able to process polynomial surfaces as well as polygonal meshes.

These preconditions are very important, since they provide the user with a wide range of possibilities to construct and extend given models. It is possible to digitize a given part such as a fender, to construct and complete a manufacturing tool within the computer and to generate the necessary output for a machine tool to manufacture this part. The complete reconstruction process of a physical part can therefore be done using appropriate software.

The basic idea of the reconstruction module is now to define a new CAD element. This element will be the *MESH* element, a piecewise linear surface consisting of triangles as its geometric primitive. Closed objects of comparable types are called *mock-up solids* and are used in several solid modeller as the basic element. The definition of the mesh element is given in detail in Section 4.1. Since the CAD

system already provides functionality to design with *exact solids*[4], the combination of these two elements leads to a *hybrid solid modelling system.* This is very important since the designer has then the possibility to model using the combination of the two representations. There are tasks such as surface generation from digitized data, which can be done easier with meshes. But there are also tasks, which are easier to solve with surfaces. The combination of both elements allows the user to choose between the representations and make use of their special properties.

A triangular mesh is chosen, because it is a standard representation used in automatic surface generation algorithm from digitized data. There are also applications such as stereolithographie or visualisation, based on triangular meshes. There are standard formats for data exchange to other systems for triangular meshes such as STL or VRML.

The task to be solved within the surface reconstruction module will now be to quickly and automatically generate meshes from digitized data and to provide the user with functionality in order to prepare, optimise and extend the model.

The first task is therefore to generate meshes. This process ideally has to be done quickly and automatically. There are examples and tasks, where the results from the mesh generation process are sufficient enough for certain applications such as copy milling. In areas, where the user does not want to deal with CAD and takes into account manual reoperation, the polygonisation is therefore the only task to be solved. For a copy milling application the reconstruction process consists of four steps; data acquisition, polygonisation, toolpath generation and manual finishing. Since polygonisation should be done fully automatically by the computer and no further optimisation and preparation is applied, the user does not need to have specific knowledge in computer aided design. The time, involved in manual reoperation, then depends on the quality of the digitized data and on the quality of the automatically generated surfaces.

As described in the requirement specification, there are many problems in digitizing. These problems might lead to plenty of rework and to long manual manufacturing times. This rework is usually unacceptable. Many of the problems can be solved using CAD packages. It is, therefore, desirable to provide functionality to optimise the digitized data set and to modify and extend the model. The tasks to be solved can

[4]Solids, defined by polynomial surfaces

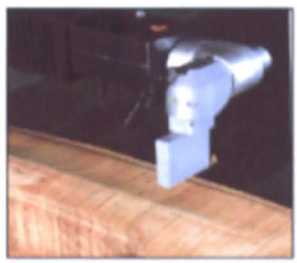
be derived from the requirement specification. There are two stages, in which optimisation can be applied to the model. First, the digitized data can be optimised. The aim is to provide the polygonisation algorithm with optimal data, which then lead to high quality meshes and which reduces the amount of time involved in the mesh generation process. The tasks for data optimisation are smoothing, spike filtering, hole filling and data reduction.

Since it is easier and less tedious to handle surfaces rather than large amounts of digitized data, optimisation functionality is also provided for meshes. The mesh preparation and optimisation functions can be derived from the functionality, given from spline surfaces. Designing comparable tools for meshes and spline surfaces is intuitive and easy to understand for the user, especially if he has CAD experience. There is functionality required for

- Alignment and modification functions to fit a digitized area into a given CAD model.
- Offset compensation for tactile measured data.
- Hole filling and surface extension in areas of occlusion.
- Surface trimming in order to generate smooth boundaries or to delete redundant surface parts.
- Splitting a closed data set in upper and lower half for die construction.
- Optimisation in terms of required storage and triangle roundness.
- Digital noise and spike filtering.
- Reconstruction and integration of features.

The reconstruction process therefore should be extended with additional functionality to optimise and repair the given data. The process of surface reconstruction within a surface reconstruction system is illustrated in Figure 17.

Additionally there is functionality required for visualisation and quality measurement. In an interactive process it is important for a user to find the areas which have to be optimised. The quality of the incoming data as well as the effect of optimisation



External data acquisition

Scan data import

- Direkt interfaces
- Native file formats
- Standard file formats

Scan data preparation

- Data optimisation
- Data repairing
- Alignment

Mesh generation

Scan data based applications

- Variance comparison
- Plane cut generation

Mesh preparation

- Alignment
- Offset compensation
- Mesh optimisation
- Mesh repairing
- Surface extension
- Surface modeling

Applications for triangular meshes

- Variance comparison
- Plane cut generation
- Visualisation
- Virtual reality
- Copy milling
- Mould, die and pattern design and manufacturing
- Stereolithographie
- Surface / Solid modeling

Figure 17: Surface reconstruction system specification.

must be measurable for both, digitized data and meshes. An optimisation process therefore can be structured into the tasks

- Detecting and visualising the problem.
- Applying an optimisation routine.
- Visualising the optimisation and comparing the quality.

Finally, it is important to keep user defined tolerances within any process of optimisation or polygonisation. The models to be generated should not only be smooth and pleasing, but they also should represent the real model within pre-defined tolerances. This, of course is difficult to guarantee. It depends on the optimisation algorithm as well as on the preciseness and completeness of the incoming data points. If the measured data points are precise, the user should be able to generate precise surfaces within small tolerances. If there are problems in data acquisition, the system should detect these problems and report this to the user rather than automatically correct them without warning. This is important, since problems with calibration or noise often result from bad calibrated digitizer. Usually, these problems can be solved by recalibration or more elaborate digitizing. The strategy here is to force the scanning systems to generate high quality data. Then the user can decide, wether to optimise the data or to recalibrate and redigitize.

With the above functionality, the user is able to reconstruct an object from digitized data, even with the problems involved in digitizing. The more optimisation functions are applied to the data, the better the result in reproduction is. With the combination of standard CAD functionality, complete models or tools can be designed, consisting of meshes and spline surfaces. It is important, that even if there is time involved in this optimisation, the reconstruction process is much faster and therefore time and money can be saved. If there is no CAD modelling requested, the model still can be reproduced. The result then depends on the quality of the incoming data. The better the data, the better the result.

Chapter 3

Literature Review

There are many ways of generating triangular meshes. First one might want to create a mesh by defining the triangles of a mesh using an interactive interface in order to *draw* a shape. There are applications, which require the generation of meshes from simple and closed polygons with holes, from surface or trimmed surfaces or from solid representations. The applications are wide spread, e.g. the generation of meshes from exact solids is necessary if the part is to be manufactured with rapid prototyping, or if finite element analysis is required. Finally the main focus within this work is on the generation of triangular meshes from scattered data points and in particular the generation of meshes from multiple range images.

Ideally one would desire to generate meshes, which approximate the surface of a real part within a user defined tolerance. The mesh should not self-intersect or overlap and it should be smooth, corners should be reconstructed perfectly and there should be no artefacts such as spikes or holes. Also one would desire to generate meshes with a low number of triangles in order to reduce the amount of storage and computing time. It is also desired to generate balanced meshes. Triangles with a large height-to-longest edge ratio (*aspect ratio*) are especially required in finite element applications. Degenerated triangles make problems since surface normals cannot be calculated which leads to artefacts in visualisation and calculation.

In the case of digitised data, an ideal automatic process is not possible for different reasons. The surface to be reconstructed is only described by a discrete set of data points and information on the surface between the data points can only be assumed. So therefore, what one can at least do is generate a mesh which interpolates or

approximates every data point within a given tolerance. Also, since this is ill defined, organise the edges of the triangles by some functional or criteria, which then leads to an ideal and well defined description of the real part.

In the literature there is no approach or algorithm given, which satisfies all the above mentioned requirements. The reasons for this are obvious. Not all the features of a part can be derived directly from the digitised data since the data only give a discrete description of the surface of the part. Features smaller than the scanner resolution cannot be detected. Areas where no data could be found cannot be reconstructed. These problems of course cannot be fixed within an automatic process. But there are also other problems in the proposed approaches, which make the algorithm either not reliable and slow or not accurate enough in our case.

In the following sections an overview on recent approaches in literature is given.

3.1 Polygonisation

3.1.1 2D-Scattered data interpolation

The idea of interpolating methods is to give a description of the area or volume by planar triangles, where every data point gets a corner point of a triangle. Clearly the quality of this piecewise linear interpolation over triangles depends on the specific ordering of edges, connecting the data points. A common criteria for triangulation is the Delaunay-criteria. A Delaunay triangulation has the property, that a circle or sphere around any triangle does not contain other points of the triangulation. This approach is called the data-independent approach since triangulation is done only in 2 Dimensions. The approach ignores the *height* of the data points, if it is used for triangulating $2.5D$ data. A description and overview of the Delaunay triangulation algorithm is given in Preparata [PS85], Guibas and Stolfi [GS85] and de Berg [dBvKOS91]. Figure 18 shows the principle of an incremental Delaunay triangulation. The point P is introduced into a given triangulation using a face split operation. Note that in Figure 39(c) the circle around the triangle V_0, V_3, P violates the Delaunay criteria since the vertex V_1 does lie within this circle. Swapping the edge E leads to triangles, which all fulfil the Delaunay criteria.

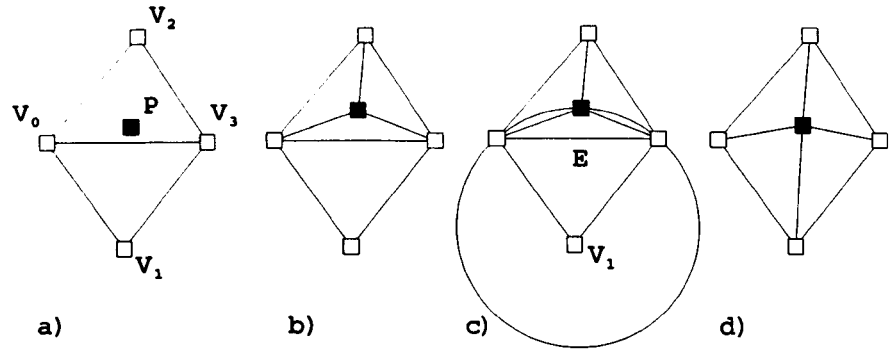


Figure 18: Triangulation using the Delaunay criteria.

It is shown by Dyn et.al. [DLR90], that triangulation quality can be significantly improved by taking the height of the data points into account. Starting with a Delaunay triangulation an improved triangulation can be found by iteratively applying the edge swap operator on the mesh. An edge is swapped, when the quality of the surface can be improved in terms of smoothness or equivalent. It was shown by Lawson [Law77], that any triangulation of a planar point set can be transformed into another triangulation by swapping edges. Different quality functions were tested and the results are compared with the original surface from which sample points were generated.

Another simple approach used in the reconstruction of objects from multiple range images is called *step discontinuity constrained triangulation.* It was described in various papers (Turk and Levoy [TL95], Hilton [Hil95], Rutishauser [RST94]). Step discontinuities occur due to occlusion in a $2.5D$ range image. All data points of a $2.5D$ range image are ordered in matrix form. Range measurements, which are adjacent in the $2D$ image plane are connected, if the $3D$ Euclidean distance between them is less than a threshold. The threshold is usually a constant, derived from the measurement resolution $\delta x, \delta y$ (Turk and Levoy[TL95]), or from the angle between the triangle normal and the optical axis (Rutishauser [RST94]). The angle criteria has the advantage, that it is independent of the resolution of the range image. In the paper of Rutishauser [RST94], the angle threshold is set to a value of 80°, which is equivalent to a distance threshold of $6\delta x$. This approach is only applicable to range images with data points, ordered in matrix form. It is simple and fast. The quality of the triangulation might be improvable, if the triangulation is optimised using the

approach of Dyn et.al. [DLR90]. Of course, this would lead to longer triangulation time.

3.1.2 Implicit surface polygonisation

A very popular approach in reconstructing a complex 3D object from multiple range images is based on implicit surface polygonisation. First each single view is triangulated using the step discontinuity constrained triangulation. In the work of Neugebauer, [NK97], [NK98], Hilton et.al. [Hil95], [HSIW96c], [HSIW96b], [HSIW96a] and Curless [Cur97] these surfaces are integrated into an 3D implicit surface representation (range image fusion). A signed distance function is defined, from which the distance of any point in 3D can be measured towards the implicit surface. Any point can also be classified *inside* or *outside* (*above* and *under*) the object. The sign is obtained by taking the direction of view of each single range image into account. In overlapping regions the signed distance measurement is done by evaluating each individual mesh and averaging the evaluated points to define the surface point. A single mesh is then extracted from the implicit surface by retriangulation methods, such as marching cubes (Cline and Lorensen [LC87]) or marching triangles (Hilton et.al. [HSIW96b]).

In a marching cubes algorithm, the $3D$ space around the objects surface is subdivided into several cubes (or voxels). Each single cube must be smaller than the smallest feature of an object but larger than the scanner accuracy. Then, the intersection points of each cube-edge with the objects surface is calculated using the above described signed distance function. Connecting these intersection points then generates a triangular mesh. This can be done very easily using a simple lookup table. Figure 19 illustrates the principle of the marching cubes algorithm.

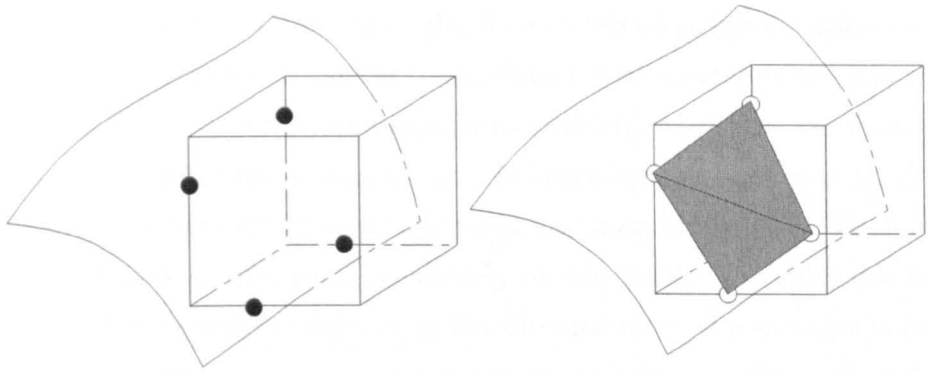


Figure 19: Principle of marching cubes.

The marching triangles approach directly generates a triangular mesh. Starting from a mesh, consisting of a single triangle generated on the implicit surface, new triangles are created at the boundary of this mesh.

New triangles are either generated by connecting neighbouring edges on the boundary of the mesh, or by calculating new vertices. A new vertex is generated by projecting a point on a circle around the midpoint of an open edge in the plane normal to the edge onto the surface of the part. Figure 20 illustrates the principle of projection. A new triangle is defined by the edge and the new projected point L_p. If this new triangle does not generate overlapping or self-intersections, or if triangles formed by connecting neighbouring edges are not better, it is added to the mesh. Special care has to be taken in order to close the mesh. The process repeats until no further triangle can be added to the mesh.

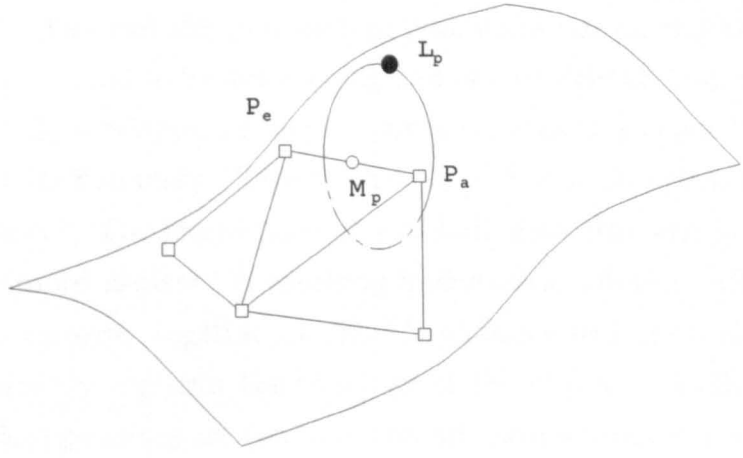


Figure 20: Generating new data points for marching triangles.

Marching cubes and marching triangles lead to initial polygonisations with a predefined size of the generated triangles. Neugebauer uses mesh optimisation and mesh refinement methods to minimise the approximation error towards the measured sample points while generating a smooth and balanced mesh. First a single mesh is generated using implicit surface representation and marching cubes. Then mesh optimisation and refinement is applied iteratively on the mesh. Triangles are subdivided if the approximation error at edges or at the circumcenter of a triangle is larger than a given threshold. The distance is measured towards the implicit 3D surface. Any triangle is subdivided using a special lookup table for the subdivision. After a subdivision step, each vertex is moved towards the center of its surrounding vertices in order to generate a balanced mesh with round triangles. The vertices of the balanced mesh are then reprojected onto the implicit surface. The process iterates until the approximation error is less than a given threshold.

3.1.3 Multiple View Combination

In the work of Soucy and Laurendeau [SL95a],[SL95b], Turk and Levoy [TL95], Karbacher and Häeusler [Kar97] and Rutishauser et.al. [RST94] the meshes from single views are merged and combined without using an implicit surface representation.

In the algorithm of Turk and Levoy, mesh combination is done dynamically by allowing sequential integration of new range images. First, triangles are eliminated in overlapping regions. Therefore the distance of any vertex of a mesh from the second mesh is calculated. If all three vertices of a triangle have a distance less than a specified offset and the plummeting points are not on the boundary of that mesh, the triangle is said to be overlapping and can be deleted from the mesh. After eliminating the fully overlapping triangles, the mesh $Mesh1$ is clipped against another mesh $Mesh2$ at its boundary. This is done by defining an *offset wall* around the boundary of $Mesh2$. The intersection of a $Mesh1$ with this wall is then calculated and $Mesh1$ is clipped against the resulting intersection polygon. After clipping, the two meshes are zippered together at their boundaries to form a single continuous surface that correctly captures the topology of the object. Finally local weighted averages of surface positions are computed on all meshes to form a consensus surface geometry. The method of Soucy and Laurendeau differs from the approach of Turk

and Levoy in the order in which integration and geometry averaging is performed.

The approach of Karbacher and Häeusler uses vertex insertion to form the consensus geometry. Initially a master image is chosen. All the other images are sequentially merged with the mesh, generated from this image. Again, each single range view is triangulated using the *step discontinuity constrained triangulation.* Then, triangles in overlapping regions are removed using an approach comparable to the one, proposed by Turk and Levoy. Additionally not only fully overlapping triangles are eliminated but also triangles, which partially overlap are removed. This limitation step leads to non overlapping meshes, which have to be connected. All data points in overlapping regions, whose absence would cause an approximation error larger than a given tolerance are integrated into the mesh by vertex insertion using the face split operator. Then, two meshes are connected using *gap-bridging* and *surface growth.* This is explained in detail in Chapter 6 since the approach derived in this work uses a comparable method. Figure 21 illustrates the principle of *gap-bridging* and *surface growth.* During the integration process the mesh is smoothed and thinned in order to reduce noise and to minimise storage. Finally the resulting mesh is optimised using geometrical and topological mesh optimisation. The geometrical optimisation process generates round triangles by moving vertices on a curved surface, defined from the generated mesh, such that the surface will get smoother. The topological optimisation reorganises the mesh using the edge-swap operator in order to generate equilateral triangles.

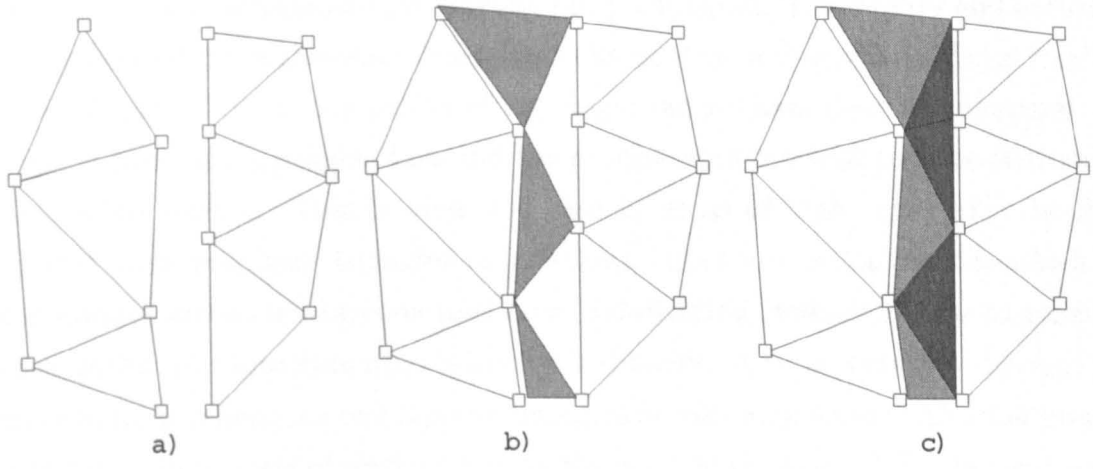


Figure 21: Two meshes, connected at their boundaries using gap-bridging b) and surface growth c).

3.1.4 Pointcloud polygonisation

There are two different approaches for polygonising an unstructured $3D$ point cloud reviewed for this thesis.

A generalisation of the convex hull of a point set in $3D$, known as *alpha shape*, is introduced by Edelsbrunner [EM94]. Hoppe et al [HDD+92] introduced a method for the polygonisation of scattered data points in 3D, based on implicit surface representation. For any data point, a normal vector is estimated by approximating a plane through neighbouring data points. The normal is equal to the plane normal. All tangent-normals are oriented consistently by comparing neighbours using a propagation method. The signed distance of a point defines the implicit surface in $3D$ to the nearest tangent plane. Then the implicit surface is triangulated using the marching cubes algorithm, introduced by Lorensen et al [LC87]. The initial triangles are then subdivided and reprojected onto the part. If the distance of the subdivision points is less than the user defined tolerance, the process is finished. During the refinement process, the resulting mesh is constantly optimised using a specific optimisation functional.

3.1.5 Discussion

The existing approaches have several disadvantages. Implicit surface triangulation does not use data points but estimated or averaged points for polygonisation. The *implicit* surface is polygonised rather than the point cloud. The quality and accuracy of the generated mesh therefore depends on the quality and accuracy of this implicit surface. If any of the triangulation of the single range views does not represent the original surface, the averaging fails and the process does not lead to a reconstruction of the initial surface. This is often the case in areas of high curvature. Meshes, generated with marching triangles or marching cubes are *initial* meshes which do not guarantee any approximation tolerance. Subdividing meshes in order to generate meshes within the approximation tolerance is difficult. It is important to prevent the algorithm from generating overlapping triangles or self-intersection. Also the process is slow for small tolerances since at corners the mesh have to be subdivided very often which leads to very high numbers of triangles. Finally the approach is very computer storage intensive, since all single meshes have to be processed at once for the implicit

surface representation.

The algorithm of Turk and Levoy sometimes fails and is therefore not reliable in some cases. To combine two meshes A and B, a 3D clipping process is applied to two slightly overlapping meshes. The intersection is calculated using a *thickened* boundary of mesh B. This wall around the boundary of mesh B is roughly perpendicular to the mesh. Mesh A is clipped against this boundary. Problems arise now in defining, how thick the wall should be. Too thin means, that no or only partial intersection can be found. Too thick leads to unwanted intersection in other parts of the mesh A. Also, and more importantly, the calculation of surface normals, which is necessary to define the wall, is difficult and unstable with noisy data. Overlapping or self intersecting walls can be the result. This leads to topological ambiguities in the clipping process and to a failure of the algorithm. Also, since first fully overlapping triangles are removed from the boundary, a mesh that is completely overlapped by another mesh is removed completely, even if it contains triangles of higher quality. Finally the mesh zippering introduces vertices generated from the mesh clipping into the mesh. These vertices are non-measured vertices. They are therefore likely not to lie on the real surface.

In the approach of Karbacher, problems arise in the reintegration approach of vertices. Reintegrating vertices using facet splits might lead to overlapping triangles or self intersections. Also, some overlapping triangles cannot be avoided in the combination step. These triangles are finally eliminated in a pre-processing step which then leads to holes or gaps in the mesh.

The approaches of Edelsbrunner and Hoppe et.al. are the most general since they can be applied to pure point clouds. The shortcoming of the approach of Edelsbrunner is the amount of storage and computing time. Only small numbers of data points can be processed. In the approach of Hoppe et.al. the calculation of the normal vector using the proposed propagation method tends to be unstable at corners or with noisy data. Again, the accuracy of the resulting surface depends on the accuracy and quality of the implicit surface.

3.2 Triangle Decimation

The main interest in methods for triangle decimation within this work is based on the sequential removal of vertices from the mesh, keeping in mind that this removal might be reversible which leads to progressive meshes. (This is explained in more detail in subsection 5.2, progressive meshes).

Schroeder et.al. [KS99] and Klein et.al. [KLS96] successively remove vertices from a mesh. The resulting hole in the mesh is then retriangulated using only existing vertices and a special polygonal triangulation method, which violates overlapping triangles. Then, the distance of the removed vertex from the new triangles is measured. If the distance is larger than a given threshold, the vertex is not removed. Vertices are also not removed, if the polygon, which has to be retriangulated, cannot be triangulated without generating overlapping triangles.

In the work of Hoppe et.al. [HDD+93], a mesh is optimised using different mesh operators. The operators are successively applied to the mesh in order to minimise a global energy functional. One of these operators is the edge-collapse, which reduces the number of vertices in the mesh. An approach, developed for the generation of progressive meshes [Hop96],[Hop97] uses the halfedge collapse operator as the basic operator for triangle decimation.

Kobbelt et.al. [CKS98b],[CKS98a] also used the halfedge collapse operator to reduce the number of vertices in a mesh. First, a binary operator decides wheather a halfedge collapse is allowed or not. Therefore, the distance of the removed vertex from the triangles, generated from the halfedge collapse is measured. Also the tangential and curvature continuity after vertex removement is measured. If the values are not within the user defined thresholds, the halfedge collapse is not applied. Vertices are removed *best first*. The quality of a vertex removement is defined by an energy functional. The local energy is calculated using a weighted function, consisting of terms to measure the approximation error, the tangent and the curvature in a mesh.

3.3 Noise Filtering

Filtering noise in digitised data, which results from the measuring system is typically removed using image processing digital filter. Low-pass, median and Gaussian filter

can be applied in order to smooth the range images [Blab], [Blaa],[Blac]. There are approaches, which use a special designed filter for range images in order to reduce blurring effects. [Nie94].

In Neugebauer et.al. [NK97], [NK98] an approach based on triangular meshes is introduced. Here a mesh is smoothed by simply *centering* a vertex within its surrounding orbit. Applying the centering operator to all vertices leads to a smoother mesh with rounder triangles. After *centering*, the resulting vertices are reprojected onto an implicit surface, described by range images. In areas, where two or more range images do overlap, the point on this surface is calculated by a weighted average from all points, generated with a signed distance function from each single range image. Averaging the measurement error between different range images using a signed distance function is an often proposed technique ([Cur97],[Hil95],[HSIW96c], [HSIW96b],[NK97],[NK98]).

Karbacher [Kar97] uses an approximation of the surface using circles. The error of a measured vertex towards this surface is denoted to be the measurement error. The vertex is then moved in the direction of an approximated normal vector towards the surface in order to filter noise or to eliminate measuring errors.

Kobbelt et.al. [Kob97] developed a discrete fairing method, which calculates new vertex positions by curvature minimisation. A discrete energy functional measuring the discrete approximation of the bending energy is minimised. The curvature (second derivatives) for every vertex in a mesh is approximated using local surface estimation.

Finally a simple approach, based on the edge-swap operator was introduced by Choi et.al. [CSYL88]. Here a mesh is smoothed by applying the edge swap operator on edges, such that the sum of the dihedral angles between adjacent triangles is minimised.

Chapter 4

Processing probed data

This chapter provides an overview on the types of digitized data (here called *digits*). The different types are compared and common properties are determined. Some optimisation and preparation can be done based on these digits. The approaches are presented and discussed below.

4.1 Data Types

Any scanning system, available on the market today, produces data points, represented by a vector in 3-dimensional space. But this is not the only information, which is provided by the scanning systems. There is more useful information obtained during digitizing. Most of this information is stored additionally in the data files. The scan records are transferred from the scanning systems via export into data files in specific, scanner dependent *native* data formats. Within these native data files, there are common and different information available for every scanning system.

Now it is senseful to process every data set equally in order to provide a common handling for all types of data. It also seems to be useful, to consider the additionally stored information in order to simplify and optimise pre-processing algorithms. Therefore it is necessary to determine the common information, provided by all scanning systems. From this definition, the data structure for digitized data can be defined.

- **Tactile measuring systems**

There are many different systems for tactile measuring available on the market.

Some of them produce data sets in a special, each of them in a common format, the NC-toolpath. NC-toolpath files consist of a header with general information and of the 3D-points. Additional information such as line segmentation is provided with *G-functions*. Data points are marked with *G08,G09*, if they are the start- or endpoint of a scan line. With this information, the segmentation of a scan record into scanlines can be calculated.

Usually tactile measuring is done on 3-axis co-ordinate measuring machines. The maximum steepness of a surface to be digitized is therefore 90^o. For this 2.5 D data, a scan axis can be defined from which every data point can be seen. This is important to notice, since all other scanning methods to be considered in this work produce data of that form. Unfortunately, this scanner orientation is not stored in the NC file, but it is possible to recalculate this orientation from the data.

The scanlines are sorted and the 2D distance of two neighboured scan lines is constant. This information can be used later in order to decide, if two scanlines should be connected or if holes remain in the data set. Again this information is not stored within the NC files but also can be recalculated afterwards. Most of the tactile measuring systems do not guarantee a constant point-to-point distance within a scanline. There is no *dense* point-cloud available, from which an object could be recalculated. There is also no *matrix* like structure available for the data from which neighbourhood information from data points between different scanlines could be defined.

- **Laser systems**

Scan data produced by laser systems are comparable to the tactile measured data sets. Any record consists of N scan lines, almost parallel to each other. A single scan line represents some sort of *plane cut* through the part within a specific resolution. The scan-direction and scan axis are always given and the maximum steepness angle for surface areas, for which data points can be obtained is 80°. This angle depends on the surface finish and colour of the part. The point to point distance within a scanline depends on the angle of the surface towards the scan-axis and is be constant on a plane. The line-distance and the point-distance are different since the common lasers produce much denser point resolution within the scan lines. The pointsets,

produced by laser systems, are dense. This allows laser data to be handled as point clouds.

- **Photogrammetry**

The data which is produced by structure lighting systems differs slightly from the above described data types. First, the data is not ordered in parallel scan lines, but in a matrix structure, which results from the pixel resolution of the camera. Therefore neighbourhood information is not only given within a scan line but also in 2 dimensions.

Also, additional information is given since most of the systems produce an additional grey scale image, from which the data points are calculated using image processing. Applying image processing tools like noise filtering seems to be interesting, but is not further investigated within this work.

Again the scan-direction and the resolution is known and can be used for further processing. Also from the resolution, the 2D distance of two neighbouring data points in the scanning plane can be calculated.

- **Summary**

From the above analysis, a data structure can be defined which is common to all data types. The primitive to be handled is a sample point in 3D. The most general structure for representing digits would be a *point cloud*, consisting of a collection of all sample points without further information. Tactile measured data could be converted into this form using resampling. But as we have seen in the literature review in Chapter 3, a fast and reliable polygonisation of these point clouds is an unsolved problem.

Therefore additional information should be taken into account in order to guarantee reliable and fast polygonisation. This additional information will be the scan direction and the ordering of the sample points in scanlines and scan records. Scanlines, generated with a common sensor orientation, are stored within such a record. They are ordered, parallel and sorted such that neighbouring polygons are stored next to each other within the record. Every scan record can be optimised and further be processed separately, taking the additional information into account.

4.2 Data Optimisation

The data provided by different scanning systems are not perfect in all cases. Every digitizing system has its specific properties and problems, for which tools are required to correct or optimise. Some of the problems are only solvable on surface level, but some can be corrected in advance at point cloud level.

One major problems with todays systems is the high number of data points and therefore with the huge amount of data, which is provided by the scanning systems. For example a car body, digitized by a range laser system, easily leads to data sets with up to 10 million data points or more. Even with the fastest computer and graphics excelerators available today, these data sets cannot be handled properly. Also, it is not necessary to process every data point, since most areas of a part are over determined. The number of data points can be reduced without loosing necessary information for describing the surface.

Optical systems have often the problem, that due to diffuse reflection at corner, single data spikes can be generated and have to be smoothed or deleted from the data sets. Figure 22 shows an example of scanlines with spikes. Sometimes the data is highly noisy due to rough surfaces or specific material. It is required to filter the noise in order to produce smooth surfaces for pre-processing. Figure 23 shows an example of noisy data from a range image.

Sometimes calibration problems lead to a constant translation error from one data record to another one. Polygonisation of these data sets would lead to noisy meshes in areas of overlapping or discontinuities when connecting two scan records. Even if the problems should be corrected by the digitizer, alignment or matching functionality is sometimes required to correct these calibration errors, especially if the object has to be reconstructed while the real model is no longer available.

Finally, it is required to remove data points in areas of no interest. Removing these data points reduces the required storage and speeds up pre-processing applications. This removal can be implemented as an interactive function, where the user defines an area (closed polygon) where all data points within this area are removed.

In the following subsections, the approaches for noise and spike filtering are described. Since alignment functionality can also be used for the fitting of digitized parts into a given CAD model, this algorithm is described, too.

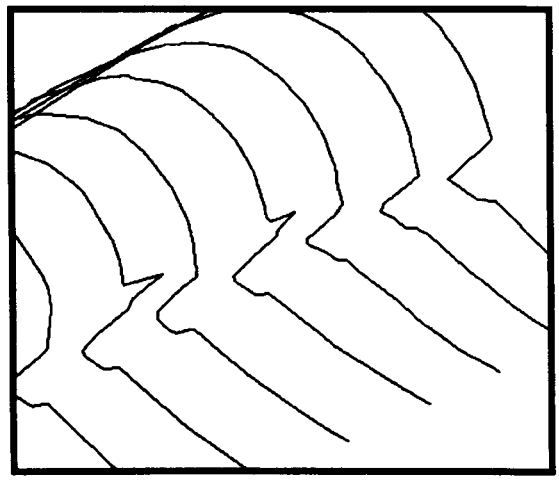


Figure 22: Digitized data from a range laser with single spikes.

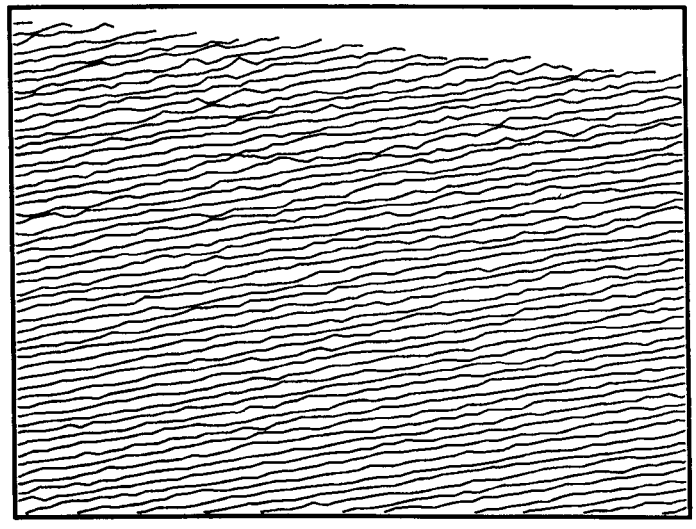


Figure 23: Noisy data from a range image.

4.2.1 Noise and spike filtering

There are many ways of removing noise or spikes in a data set. The most important problem is to filter the data points without blurring features such as corners. Since our data structure consists of scanlines (polygons) rather than point matrices or point clouds, we only consider 1D filtering of the polygons at this level of optimisation.

Data or signal filtering is discussed in the literature for signal and image processing ([Blab]). Many approaches are based on the *Fourier* transformation of a signal

into the frequency domain. Then, appropriate highpass, lowpass or bandpass filters are applied in the frequency domain. Lowpass filters attenuate the high frequency components while passing the low frequencies within a specified range. Highpass filters do exactly the opposite to lowpass filters. It attenuates the low frequency components while passing the high frequencies. Bandpass filters only allow those frequencies within a specified range. In this sense, lowpass and highpass filtering are just special types of bandpass filters. After filtering, the signal is retransformed from the frequency domain by an inverse Fourier transform [Blab].

Filtering in the real domain is based on some form of moving window principle. A sample of data about a given element of the signal is processed giving (typically) one output value. The window is the moved to the next element of the signal and the process is repeated. A common real space filter is the Finite Impulse Response filter (FIR) of the form

$$S_i = \sum_j P_{i-j} F_i$$

where F_i is the input, S_i is the output and P_i is the *kernel* of the filter. A kernel consists of a finite number of $j = 2N+1$ non-zero coefficients. The most simple kernel with just three coefficients is the $P = (1/3, 1/3, 1/3)^T$ filter. This filter averages a data point considering its direct neighbours.

Before defining the appropriate kernel for filtering, the character of the noise has to be determined. It is expected, that the noise distribution is Gaussian for which a Gaussian filter could be applied in order to smooth the data. Therefore, a highly accurate plane has been digitized using different scanners. Then, an optimum plane has been fitted to the data points and the distance of any data point from the plane has been calculated. An example of such a digitized plane is given in Figure 24. The data set was produced by a range laser and consists of 478000 data points. The maximum distance of any point from the plane is less than $0.1mm$. Figure 25 shows the resulting noise distribution. This is the distance of any point from an optimum plane through all data points. As expected, the noise distribution is Gaussian and the appropriate Gaussian filter can be applied.

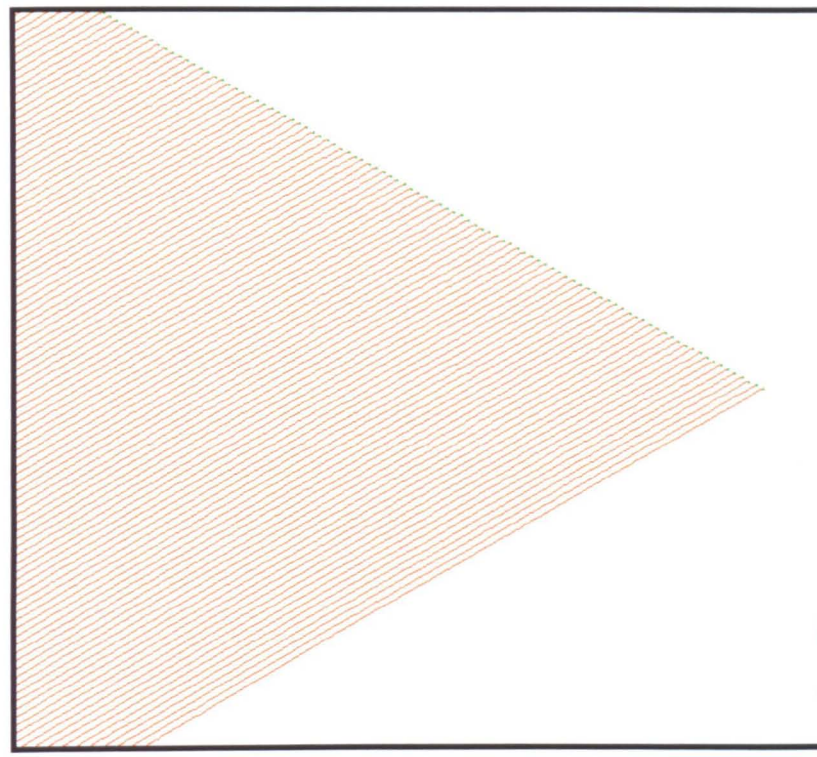


Figure 24: Highly accurate plane, digitized with a range laser.

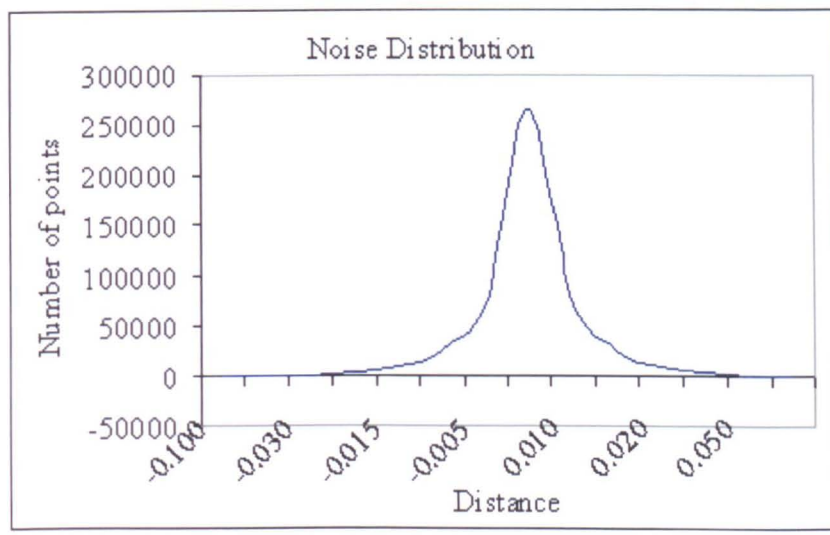


Figure 25: Noise distribution.

Filtering any point of a scanline would blur the corner in a data set. It is, therefore, important to detect the corner in advance and to leave them fixed in the filtering

process. A simple method of corner detection in a data set has been evaluated to be best suited for our case. A polygon is simply filtered with the Gaussian digital filter and all data points are marked, where the angle between neighboured lines is greater than a threshold of 15°. Then, the nonfiltered polygon is filtered again considering the corner points which are left fixed. Clearly not only sharp corners but also small radii are detected by this method which is desirable for our study. Spikes are detected with another simple method. Again the polygon is filtered once with a simple finite impulse response (FIR) filter. Then, the distance of any filtered point from its original position is measured. If this distance is greater than a user specified tolerance, the point is detected to be a spike and can be removed or filtered.

There are more sophisticated methods of noise filtering and corner detection discussed in the literature. A good introduction and overview into the field of image processing and signal analysis, including the problem of digital filtering and corner detection is given in [Blab] and [Blaa].

4.2.2 Alignment

Alignment is required for two different reasons. Firstly, it is sometimes required to align two partially overlapping scan records, if they do not match properly due to calibration problems. Secondly alignment of partial changes into a given CAD model is one of the tasks to be solved for a reconstruction system. This is by far the more important task, since it saves a lot calibration time whereas calibration problems should always be solved within the scanning system.

There are different algorithms described in literature for alignment. The present approach is based on the algorithm, proposed by G. Turk and M.Levoy [TL95]. Here only two scan records at a time are registered[1]. The two elements are the element to be aligned and a target element, which is left fixed in the alignment process. The target element can be a scan record, a mesh or a CAD surface. The alignment is done in two stages:

- Rough registration of two scan records with user interaction.
- Automatic distance minimisation using an iterative, closest point algorithm.

[1] In the literature, the process of alignment is also called *registration* or *matching*, which defines roughly the same task.

In the first step, the user defines interactively corresponding points on the scan record to be aligned and on the target element. Then a first transformation is calculated from these selected points using a closest point algorithm, which leads to a first, rough alignment of the scan record.

Secondly an iterative algorithm calculates the best transformation in order to minimises the distance between an arbitrary point on the scan record and the target element. This is done in the following stages:

- Select sample points on the scan record.
- Define corresponding points on the target element. A corresponding point is the point with the shortest distance from the sample point.
- Calculate the optimum transformation using the closest point algorithm.
- Repeat steps 2 and 3 until the mean and maximum distance of the sample points cannot be further minimised (or is smaller than user defined tolerances).
- Repeat all steps with a larger amount of data points until all sample points on the scan record are taken into account or the user stops the iteration.

The aim of the algorithm is to calculate quickly a first transformation using a small number of data points. This transformation is usually very good and sufficient for an optimum alignment. The user then has the choice judge the actual transformation and to stop the process, if the result is satisfying. This reduces the calculation time and large data sets can be aligned within acceptable times.

The optimum transformation is calculated by a weighted least squares minimisation of the point-distances, using the following functional:

$$E = \sum_{i=0}^{n-1} w_i |A_i - R(B_i - B_c) - T|^2$$

where T is the translation vector, which is calculated from the difference of the midpoint of the sample points A_i and the midpoint B_c of the target points B_i, and R is the rotation vector. R is calculated using a cross-covariance matrix. A precise description of how to calculate the rotation vector is given in [Hor87] and [BM92]. w_i are weights, defining the quality of the corresponding points within a range of $0, ..., 1$.

The quality is defined by the angle between estimated normal vectors on the scan record and on the target element. The smaller the angle between the two vectors, the larger is the confidence to have found equal points and therefore the larger is the weight.

Some of the sample points should not be taken into account. If scan records only overlap partially, which is usually the case, sample points which do have corresponding target points on the boundary of the target element, are not taken into account. Also the user can deactivate sample points. If partial changes have to be fitted into a given model, the user selects only sample points for the alignment in areas, which have not been modified and therefore should fit.

In order to speed up the alignment process, only a few sample points (typically 20) are taken into account in the first iteration. In the following iteration steps, the number of data points is doubled until all the points are taken into account. The selection of data points is done hierarchically, in that only every second or third data point is taken into account.

4.2.3 Discussion

The algorithms used for optimisation manipulate the data in order to generate a data set, which fits better to the real surface of a part. Since data is manipulated, this can only be done if the error is acceptable to the user. Noise and spike filtering as well as alignment seems to be a task for the digitizing systems, since these are problems which are specific to every scanner. Also during measurement, there are more possibilities in correcting problems with remeasuring specific areas or recalibrating the system. Except for data reduction, all this functionality is seen as correcting the problems of the digitizing systems. Since today's systems do not always provide perfect data sets, this functionality must be given to the user of a surface reconstruction system. All functions should be handled with care since data points are manipulated and the error with the real surface might get larger. Also, there are approaches for noise filtering based on triangular meshes. Filtering meshes have the advantage, that 2D neighbourhood information rather than 1D is available. Therefore, not only within a scan line but within the surface, filtering can be applied. This leads to much better results and should be applied at that stage if possible.

4.3 Results

To conclude the chapter, two results obtained from the proposed algorithm are given. First, an example for alignment is given in Figures 26- 28, which illustrate the process of integrating partial changes into a given model. Figure 26 shows a given CAD model with a hole (green) and a digitised object (red), which defines the geometry of the missing area. Figure 27 shows the example after rough alignment. The alignment was calulated from three corresponding points on the source and target element. These points were defined interactively by the user. The result after the iteration process is given in Figure 28. Note, that the source and target element nicely overlap. The mean deviation error of all sample points in this example is less than 0.03mm.

In the second example, the results from noise filtering are shown in Figure 29-33. Figure 29 and Figure 30 shows the example without filtering. The object is not smooth and the shading has artifacts. Figure 31 shows a comparison of the scanlines before (blue) and after (yellow) noise filtering. In Figure 32 and 33, the result after noise filtering is shown. The object is now smooth and visually pleasing.

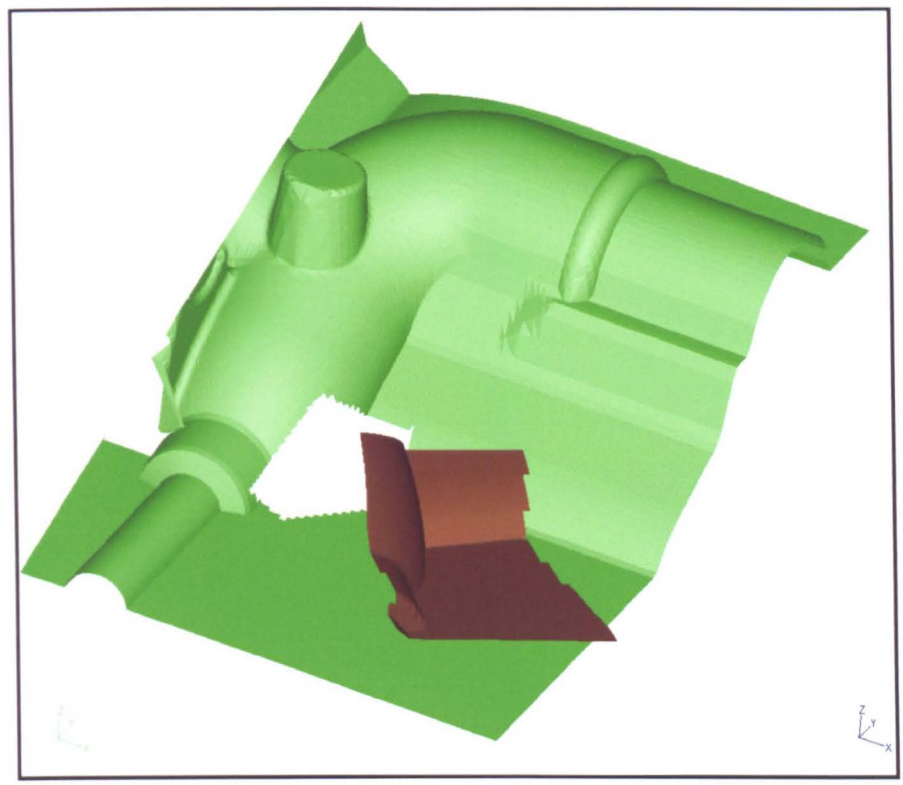


Figure 26: Alignment part I of III
Target element and element to be aligned.

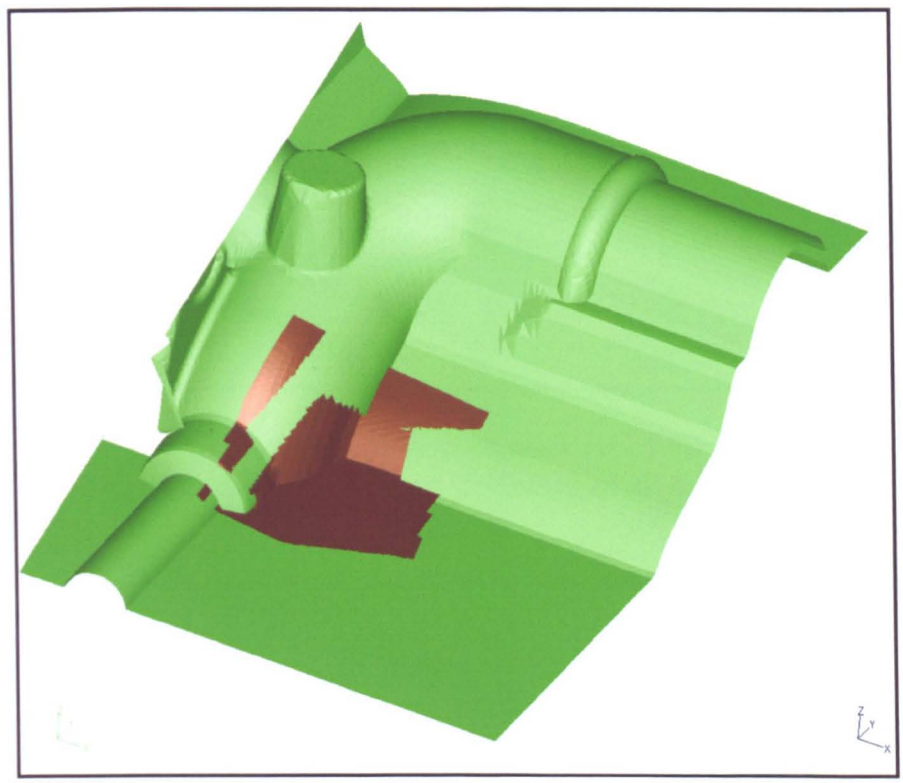


Figure 27: Alignment part II of III
Rough alignment, achieved with user interaction.

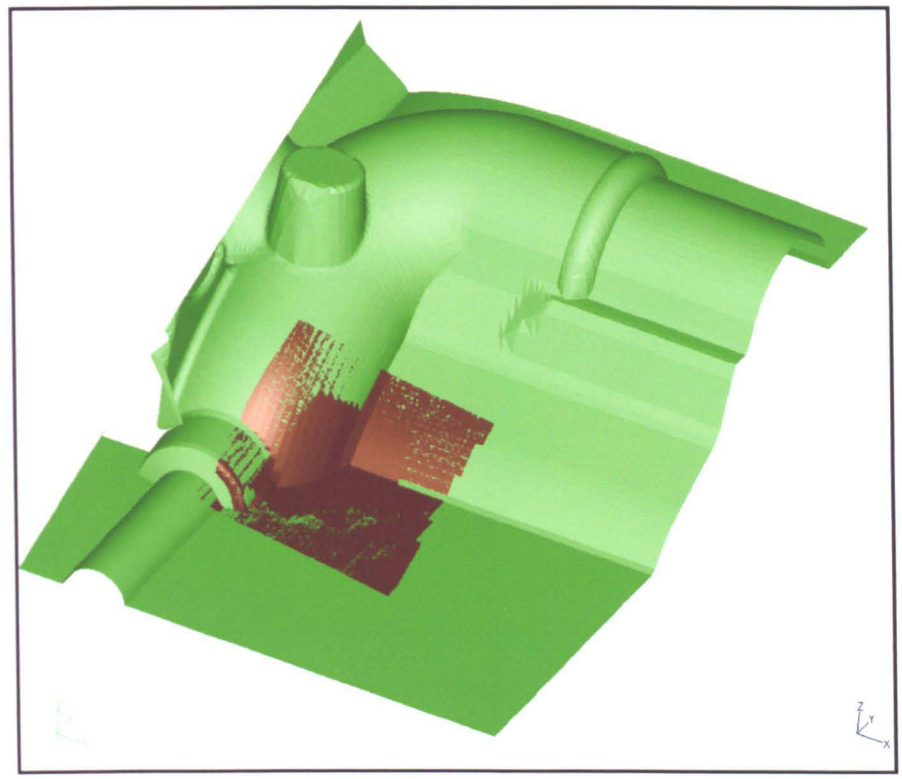


Figure 28: Alignment part III of III
Example after full alignment. The mean deviation of all sample points is less than 0.03mm.

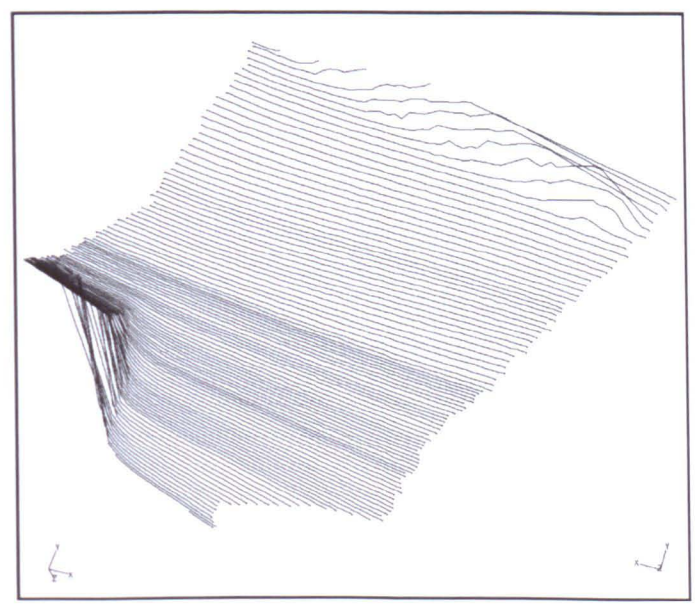


Figure 29: Noise filtering, part I of V
Here the scanlines to be filtered are shown.

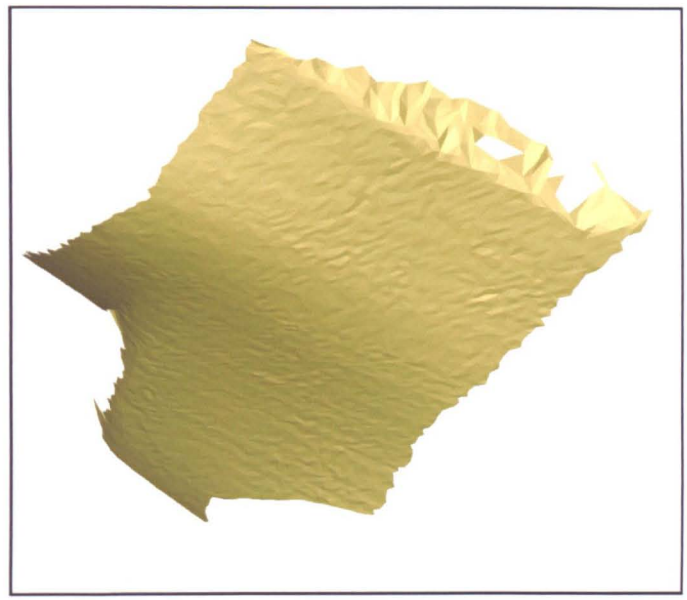


Figure 30: Noise filtering, part II of V
Resulting mesh after polygonisation without filtering the scanlines.

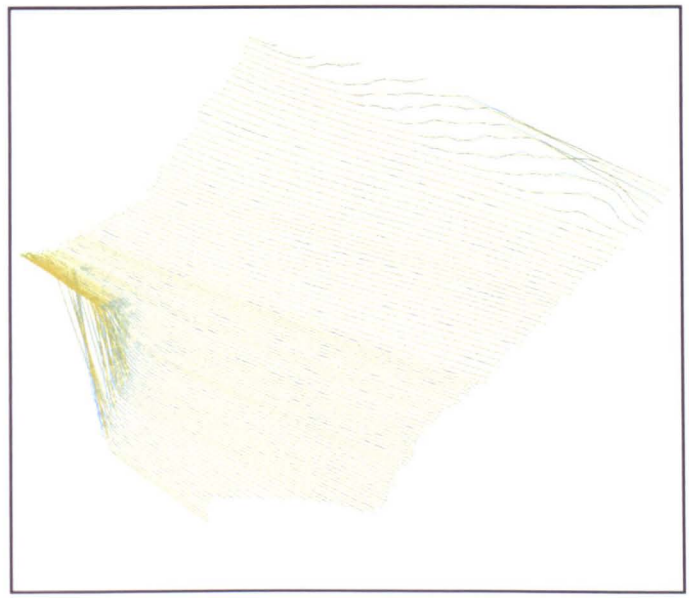


Figure 31: Noise filtering, part III of V
The scanlines before filtering are shown in light blue. The filtered scanlines are shown in yellow.

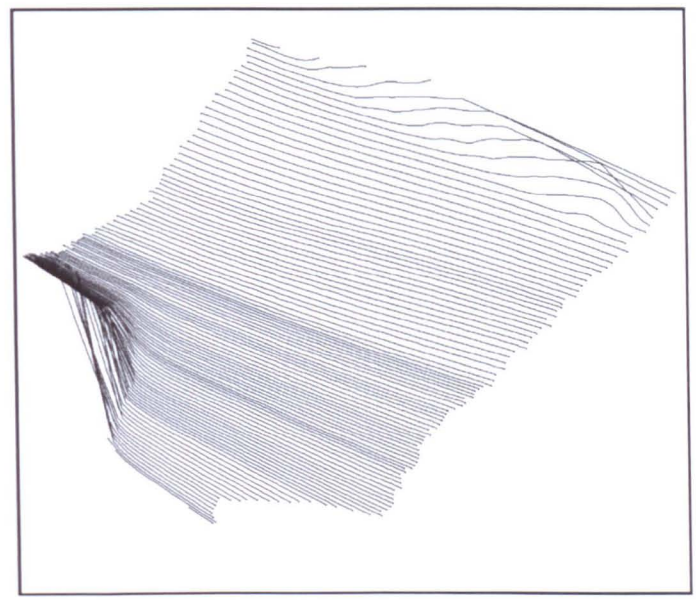


Figure 32: Noise filtering, part IV of V
Filtered scanlines.

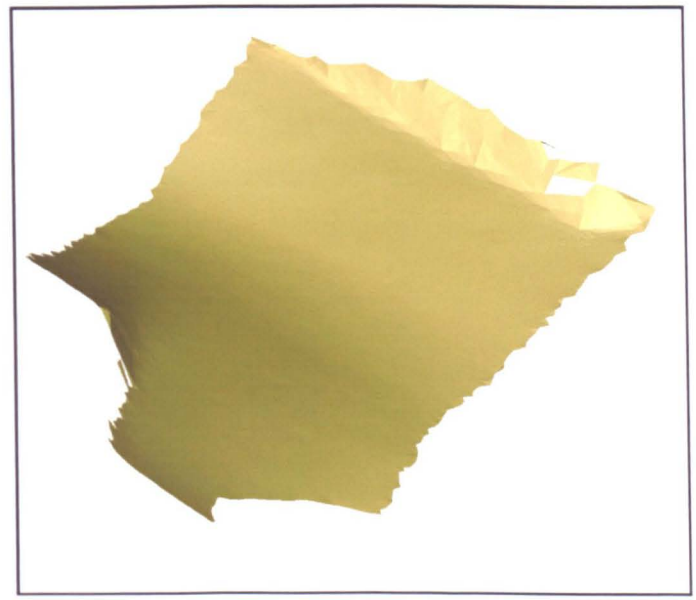


Figure 33: Noise filtering, part V of V
Resulting mesh after scanline filtering and polygonisation.

Chapter 5

Data Structure

From the digitised data, prepared and optimised with the functionality described in Chapter 4, it is desired to generate a continuous surface, which from the requirement specification should be a triangular mesh. This *mesh* element will be a new CAD element, which can be displayed, saved in the data base and which is the basis for optimisations and applications. There are different representations for polygonal meshes described in the literature. The most impressive overview on data structures is given in Ashdown [Ash], which provides more than 50 references. The meshes to be generated should also be compatible to applications such as stereolithographie or finite element analysis and they should be transferable via standard interfaces.

In this Chapter, the data structure for an efficient representation of triangular meshes is derived. Also some of the topological operators used in further applications are described.

5.1 Triangular Meshes

The data object to be defined has to satisfy some specific requirements, which can be derived from the requirement specification and from the requirements due to the applications to be developed. Of course the geometry of an object has to be represented by this mesh. Geometry is given by the 3D co-ordinate information of the sample points. Topological information is also important for many applications. In many applications it is necessary to *navigate* through the mesh, i.e. from one triangle to the neighbouring one or around the boundary of a mesh. It is, therefore, important to

determine adjacent triangles and to calculate any triangle around a triangle corner. Finally, and most importantly, the object to be modelled must be *realisable*, which means that the object must be constructable in a three-dimensional space. Since we are only interested in realisable models, we restrict our models to plane models of 2-manifolds. A 2-manifold is a topological space where every point has a neighbourhood topological equivalent to an open disk of E^2 [Män88].

A polygonal mesh consists of vertices, halfedges, edges, loops and facets. Vertices represent the geometric information. Halfedges, loops and facets represent the topology, edges or neighbourhood information stored in every halfedge define the connectivity. Facets define the surface regions of the object. Every facet consists of N loops. Every loop is defined by a number of halfedges. Halfedges are line segments defined by an index or pointer to a begin and end-vertex.

In order to guarantee topological identification it has to be guaranteed, that any halfedge should be adjacent and identified with exactly one other halfedge in the mesh, both described by the same vertices (except those at the boundary of a mesh, which do not reference any halfedge at all). Figure 34 shows an example of a triangular mesh with *hanging nodes*. The mesh shown on the left side is not allowed and topologically not closed. On the right side the topologically correct mesh is shown. This rule also has to be fulfilled in order to transfer triangular meshes via the STL interface.

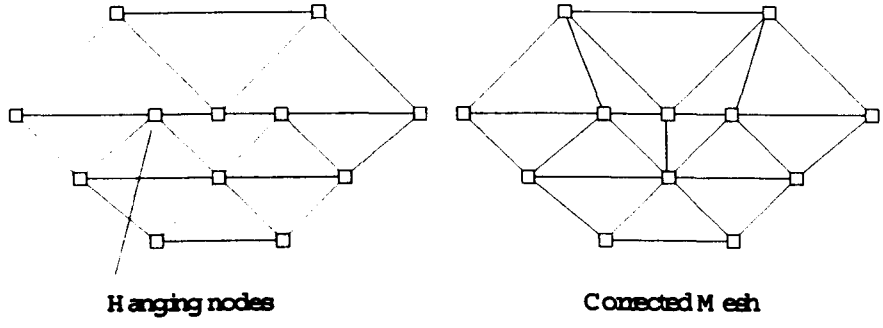


Figure 34: Mesh with *hanging nodes*

Secondly, for each collection of identified vertices, the loops identified at that collection can be arranged in a cycle such that each consecutive pair of polygons in the cycle is identified at an edge adjacent to a vertex from the collection (*surface subdivision*) [Män88].

The orientation of the triangles is important, too. All triangles should be oriented

consistently, which means that the normal vector of neighbouring triangles should point into the same direction. More precisely, a plane model is orientable, if the direction of its loops can be chosen so that for each pair of identified edges, one edge occurs in its positive orientation in the direction, chosen for its loop, and the other one in its negative direction [Män88]. This condition is known as the *Möbius rule*. The *Klein bottle* (Figure 35) is an example of such a not orientable model.

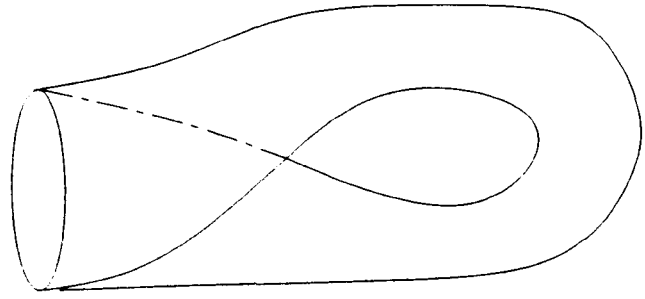


Figure 35: A twisted cylinder, topologically equivalent to the *Klein bottle*, which is not orientable.

Realizability and orientability guarantee that the model to be constructed is *topological consistent*. Many problems arise if consistency is not guaranteed. Navigation is sometimes not possible and it is not possible to define inside or outside for objects such as the Klein bottle and most importantly the object cannot be manufactured. Both rules have to be guaranteed when building up or manipulating an object. Figure 35 shows an example of a Klein bottle. Note, that the surface orientation changes at the top of the bottle. Figure 36 shows an example of a solid with non-manifold surfaces, which can also not be manufactured. For more information the reader is referred to the book of M. Mäntylä ([Män88]).

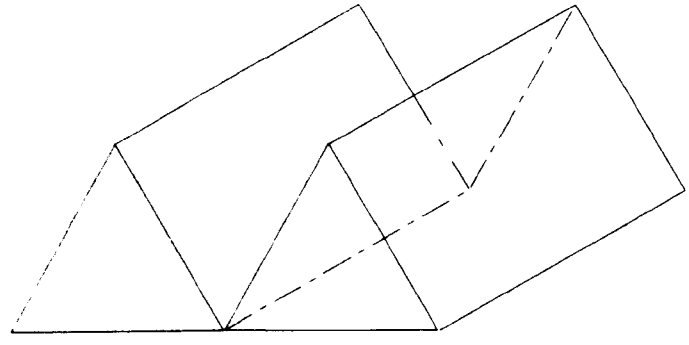


Figure 36: Solid with nonmanifold surfaces.

5.1.1 Surface tessellation

Kobbelt ([Kob98a]) used a simple representation for triangle meshes in his work. Here a mesh consists of a list of vertices and a list of triangles (facets). Vertices are given by their $V(x, y, z)$ co-ordinates. They are stored in lists and can be identified by their ordering position. A facet F consists of pointers to its three corner vertices V_0, V_1, V_2. The vertices are ordered clockwise and the connection of two vertices define a halfedge of the facet $H = 3 * F + C, C \in \{0, 1, 2\}$. Since the number of halfedges is equal to $3 * F$, the facet index can be calculated from $F = H/3$ and the halfedge index $C \in \{0, 1, 2\}$ within the facet from H mod 3. Three more pointers F_0, F_1 and F_2 are stored which define neighbouring faces. The neighbourhood information between adjacent facets can be calculated from the shared vertices being referenced by adjacent triangles. Finally a pointer or index for each vertex V to one of its adjacent facets F is required in order to identify all facets around a vertex. Figure 37 illustrates the data structure.

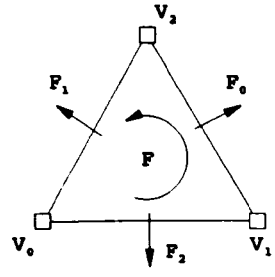


Figure 37: Surface tessellation.

Only one of the adjacent facets has to be identified, since all others can be calculated from the neighbourhood information using navigation routines. Therefore it is required to store the following data:

```
struct vertex {
  double X,Y,Z; // co-ordinates of a point in 3D
  int    F    ; // index to one of its adjacent facets
}

struct facet  {
```

```
    int V0,V1,V2; // index to three vertices
    int F0,F1,F2; // index to neighbouring facets (alternatively
                  // index to neighbouring halfedges H0,H1,H2).
}
```

Vertices and facets are stored in lists. Facets and vertices can then be accessed directly with a simple interface. Navigation can also be done using the facet pointer from a vertex, and using the facet pointer to neighbouring facets.

The surface tessellation representation is minimal in terms of data storage. On the other hand, direct access to the *full* edges in a mesh is not possible since they are not stored within the mesh. Also, *complex* vertices cannot be handled. Let a *fan* be all neighbouring and connected facets around a vertex. A vertex is complex, if there is more than one fan adjacent to a vertex. Since there is only one reference from a vertex to an adjacent facet and different fans do not reference each other, only one fan can be referenced. Complex vertices are not desired in the processing of meshes, but they are sometimes not avoidable, when e.g. triangles have to be deleted in an arbitrary order or if two meshes have to be connected. Figure 38 shows an example of such a complex vertex.

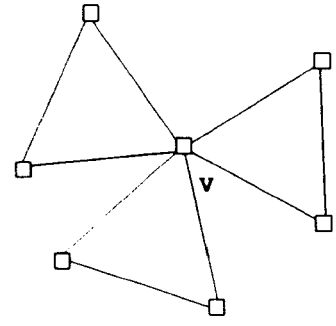


Figure 38: Three facets, meeting at one common *complex* vertex V.

5.1.2 Boundary representation (B-Rep model)

Another approach can be derived from the *winged edge* or *half-edge structure*. It is introduced in the area of solid modelling and represents the main solid representation for the solid modeller, proposed by [B.74]. The basic idea is to represent faces by cyclic lists of edges (loops). These edges are line segments, which are connected by

pointer to the next and previous edge. The connectivity is described within these edges. They store the connection of two neighbouring facets. A detailed explanation and description is given in [Män88], Chapter *boundary models.*

One advantage of this data structure is, that not only triangular meshes, but meshes with arbitrary n-gons or even halfedges consisting of polynom curve segments can be represented. This boundary representation leads to a full representation of solid models (B-Rep model). A solid is initialised and modified by *Euler* operators. With these operators, vertices and halfedges can be defined, modified and deleted. The disadvantage of this representation is the amount of storage needed when it is reduced to triangular meshes. Every face consists of only one loop and every loop has exactly three halfedges. A generalisation does not seem to be useful. Also the algorithms to handle the object are more complex. If only Euler operators are used, the representation is complicated to initialise. On the other hand, using Euler operators it is guaranteed, that the topological consistency is not violated.

5.1.3 Edge-based representation

In this work, a data structure is introduced which is closely related to the above models. It is an edge-based representation. The connectivity of adjacent triangles is stored in an edge data structure. Edges contain two pointers or indices to two adjacent halfedges, and two pointers or indices to its corner vertices. If the second index to a halfedge is -1, the edge is at the boundary of the mesh. These edges are called *open* edges in contrast to *closed* edges inside the mesh. Any facet consists of indices to three vertices and three edges. A vertex consists of the co-ordinates of the point in 3D and of N references to edges. We extended the vertex data structure proposed by Kobbelt in order to handle the case, where two or more complices meet at only one vertex. In every vertex we therefore store an index to one edge for every complex, adjacent to the vertex. As an optimisation for navigation, this edge index must be a specific one for an *open* vertex at the boundary of a mesh. When collecting all edges around this vertex in a counter-clockwise order, the first and last edge is open. The edge-index, stored in the vertex must be the index of the end edge from this collection. This allows simple, fast and easy navigation since any complex can be collected starting from the edge stored in the vertex and rotating counter-clockwise

around the vertex.

Therefore we define the following data structures:

```
struct Vertex {
  double X,Y,Z; // Vertex co-ordinates
  int    N;     // Number of complex
  int   *E;     // N indices to any complex
}

struct Edge {
  int V[2]; // Two pointer to its corner vertices
  int H[2]; // Two pointer to adjacent halfedges,
            //   H=3*F+C, C=0,...,2
}

struct Face {
  int V[3]; // Corner vertices
  int E[3]; // Adjacent edges, from which neighbours can be derived.
}
```

There are three lists or arrays necessary to store the vertices, edges and facets. Additionally we need one more array for storing a sorted list for accessing the edges at complex vertices. If a vertex is not complex, we directly store the index to the edge in a vertex, rather than extending the complex array.

Using the edge-based data structure it is possible to directly access vertices, facets and edges. There are many applications, which can be formulated very easily having edges accessible directly. The data structure can be extended to facets with more than three corner vertices and it is straightforward to implement. To guarantee that navigation is always possible and that realizability and orientability are always given, there are certain rules which have to be guaranteed when building up the data structure:

- The three corner vertices of a facet F must have different indices ($V_0 \neq V_1, V_2$, $V_1 \neq V_2$).

- The three edge indices of a facet F must refer to different edges ($E_0 \neq E_1, E_2$; $E_1 \neq E_2$).
- The vertex and halfedge indices of an edge must be disjunct ($V_0 \neq V_1$; $H_0 \neq H_1$).
- Any H_0 of an edge E must be a value of $H_0 \in [0, 3*Fnr-1]$, Fnr is the number of facets in a mesh. H_1 additionally might be -1, if E is a border edge.
- Not more than two facets must refer to an edge E in a list. This facets must be neighbouring, having V_0, V_1 as common vertices.
- Any facet F must refer at its halfedge $H = 3*F+C$ to an edge $E \in [0, Enr-1]$. The edge itself then must refer either with H_0 or H_1 to exactly this halfedge H. Enr is the number of edges in a mesh.
- Any vertex V points to an edge E. E must have V as its beginning or endpoint.
- If the cycle of edges around the vertex V is open, the edge index $E[N]$ must be the beginning edge of this cycle.
- The edge index E of a vertex V must refer to an edge $E \in [0, Enr-1]$.

5.2 Progressive Meshes

In computer graphics applications it is often required to handle highly detailed objects which require a high amount of storage and computing time. In order to improve the performance further it is common to use a detailed mesh, when the object is close to the viewer, and a coarser approximation as the object recedes (*Level of detail* approximation). The *progressive mesh*, introduced by Hoppe ([Hop96]), gives an continuous-resolution representation of a mesh with a minimum of required computer storage.

For further explanations, two mesh operators, the *edge-collapse* and the *vertex split* have to be introduced. The halfedge-collapse operator is used to reduce the number of triangles in a mesh. It is important to notice, that no additional geometric information is introduced with this operator and that this transformation is sufficient enough for

simplifying a mesh. To reverse the edge-collapse, a vertex-split can be applied. A more detailed description of the operators is given in subsection 5.3.2 and 5.3.3. An initial mesh can then be transformed into a more simplified representation by applying a sequence of edge-collapses. These collapses are stored in a list. The particular sequence is chosen such that it determines a good smoothness of the decimated meshes (This is explained in more detail in Chapter 7, mesh optimisation). Different level of details can then be generated by applying edge-collapses to reduce the number of triangles, or vertex-splits in order to increase the number of triangles in the mesh. These operations, once the progressive mesh is initialised, are very fast and can be applied at runtime. Therefore, the number of triangles for visualisation can be chosen such that graphic-performance and required storage can be minimised.

Figures 39-42 show examples of a mesh at different level of detail. The number of triangles in Figure 39 is 227644. Figure 40 shows the object with 94086 triangles. In Figure 41 the number of triangles has been reduced to 4983. Note that even with only 4983 triangles, the object contours are still visible. From a far point of view, the object can still be recognised and identified.

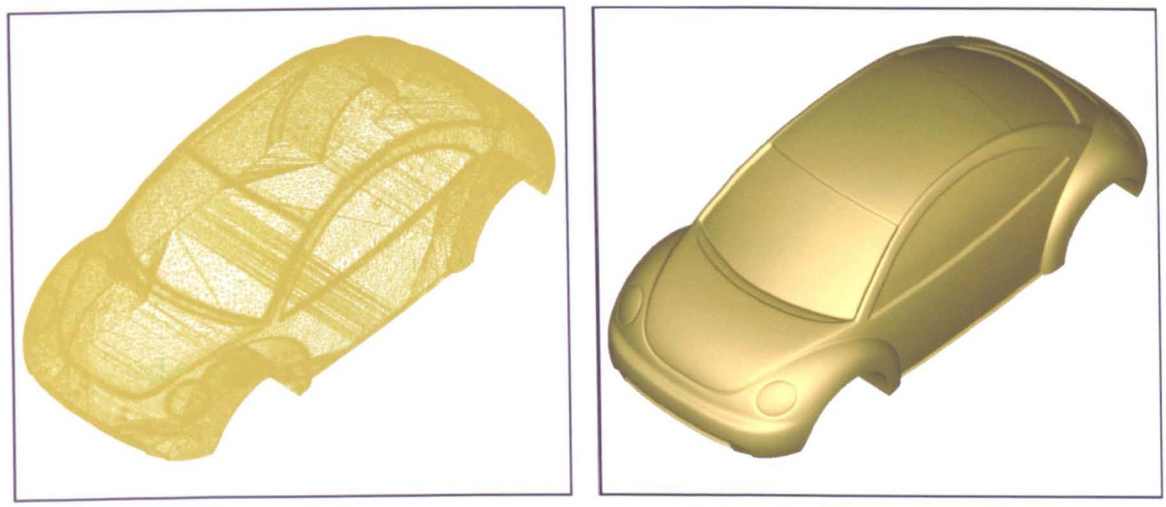


Figure 39: Mesh consisting of $Fnr = 227.644$ facets.

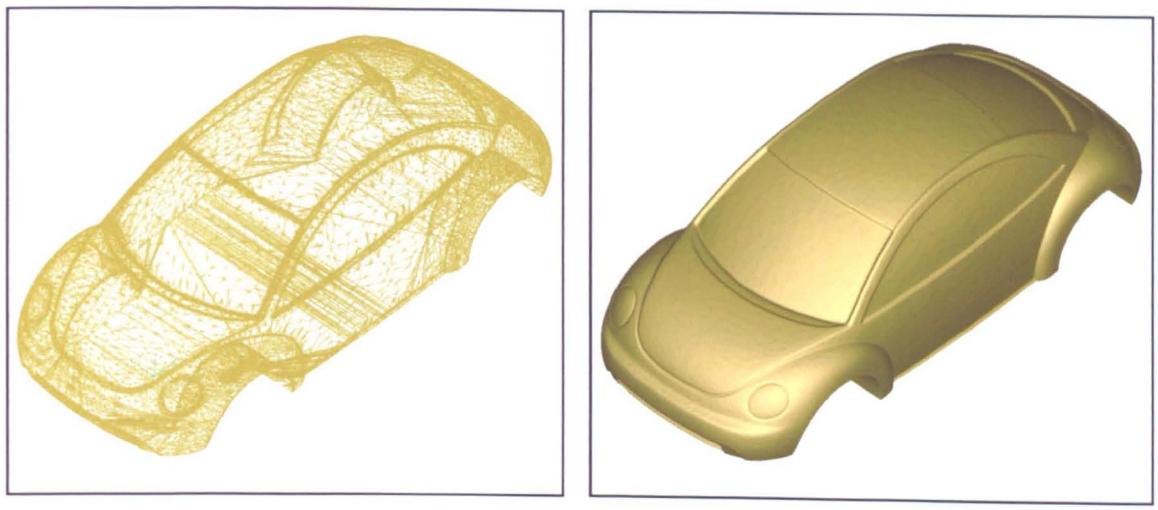


Figure 40: Number of facets reduced to $Fnr = 94.086$.

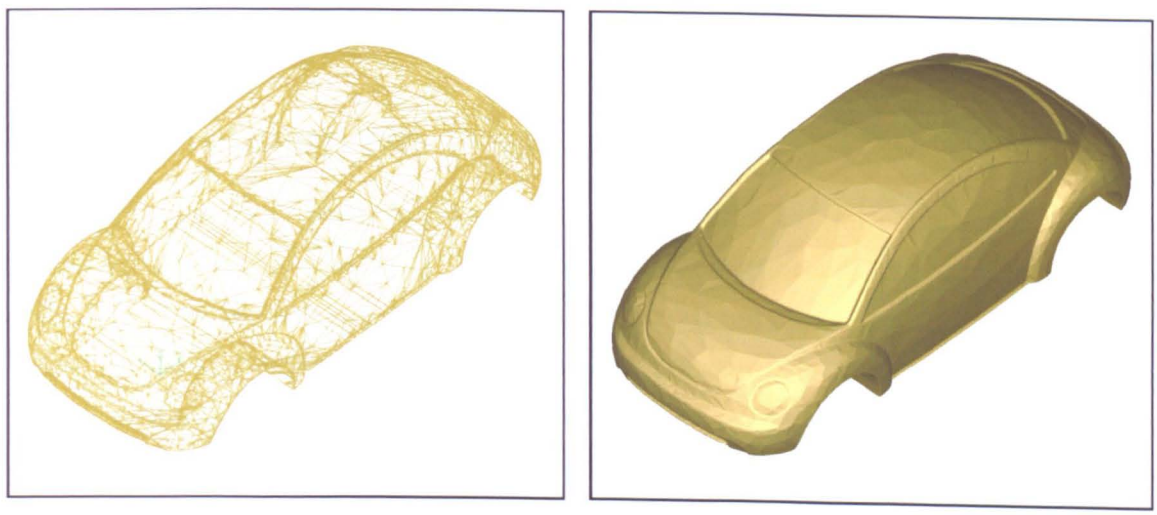


Figure 41: Number of facets reduced to $Fnr = 20.200$.

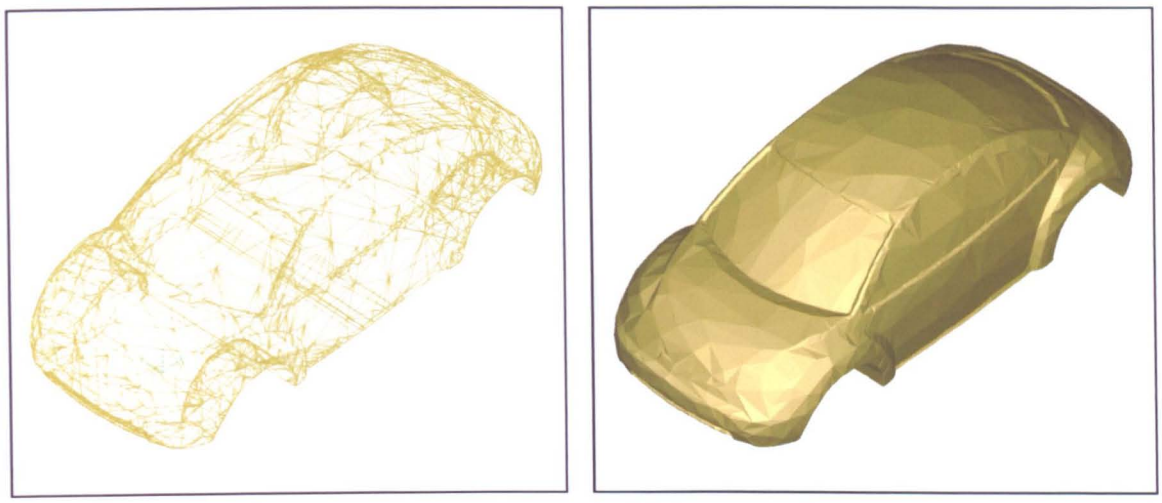


Figure 42: Number of facets reduced to $Fnr = 4.983$.

5.3 Mesh Operator

5.3.1 Fans, Stars and Orbits

In all our applications for triangle meshes, it is required to find neighbourhood information for a given facet, edge or vertex. The neighbourhood of a vertex can either be facets, edges or vertices. A *fan* is denoted to be all neighbouring triangles around a vertex, sorted and ordered counter-clockwise. Collecting all the edges with the vertex as their beginning or end point, all sorted and ordered counter-clockwise, leads to a data structure we call a *star*. Finally the *orbit* of a given vertex V consists of all neighboured vertices V_i of V, again sorted and ordered counter-clockwise. Additionally the orbit is extended such that the edges E_I, connecting neighbouring vertices V_i, V_{i+1} are stored, too. Figure 43 illustrates the different structures.

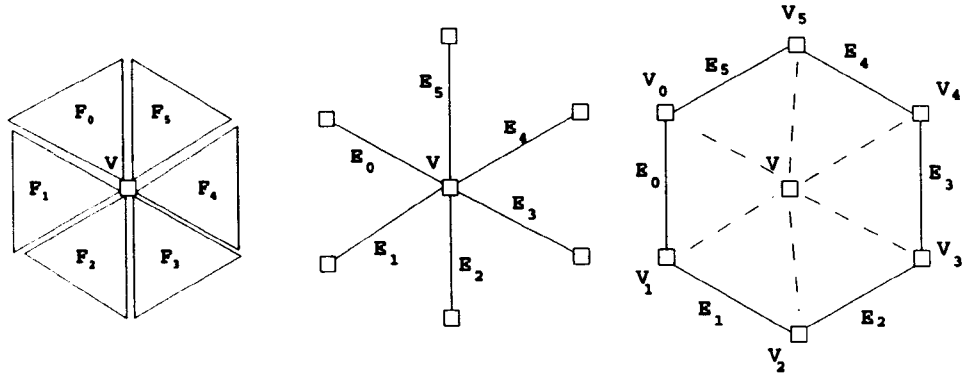


Figure 43: Fan, Star and Orbit.

Fans, stars and orbits are obtained by rotating around a vertex, starting from the edge which is associated with the vertex. The first edge can be directly obtained from the index, stored in V. The edge E_0 then has to be normalized such that the halfedge H_0 of E_0 has V as its beginning vertex. The next edge in counter-clockwise order can be obtained from the next edge on the facet, referenced by H_1. Again the rule is to rotate counter-clockwise around the facet. The edge index can therefore be obtained by the rule $C = H \bmod 3$, $E_{I+1} = F.E(C-1)$. Fans, stars and orbits can either be closed or open at the boundaries of the mesh.

5.3.2 Edge-collapse (or halfedge-contraction)

The edge collapse operator is used for removing vertices in a mesh. Let P and Q be the begin and end-vertex of a halfedge H. For the general collapse operation, the vertices P and Q collapse into an arbitrary vertex R. Halfedge collapse means either $R := P$ or $R := Q$. When pulling P into Q, all adjacent triangles to P have to change their vertex reference from P to Q.

Unfortunately the collapse operator can cause topological inconsistencies. It is not allowed to apply the operator if the resulting mesh consists of equal facets or double edges. There are two rules, which have to be satisfied in order to fulfil topological consistency:

- There have to be at least four neighboured triangles at each vertex before the collapse (Valence of (P and Q) > 3)

- There should be no common vertex in the orbits of P and Q (except P, Q itself and $R1$ and $R2$).

For each halfedge-collapse, two triangles, three edges and one vertex are removed from the mesh. (One triangle and one edge less at the boundary of the mesh). Figure 44 shows the principle of the operator.

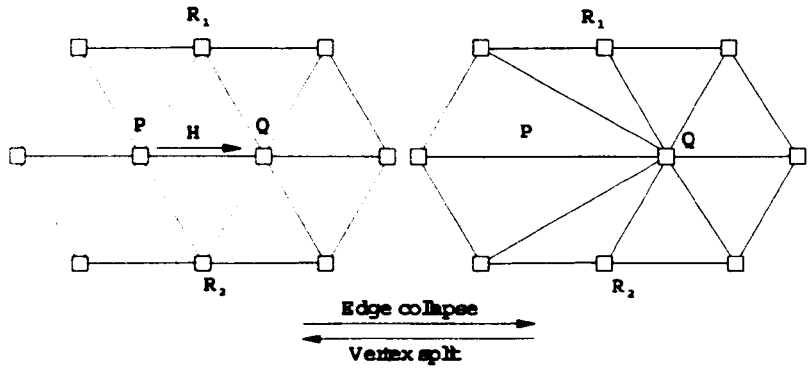


Figure 44: Edge-Collapse and Vertex-Split

5.3.3 Vertex split

Splitting an edge E at a new arbitrary vertex V is called vertex-split. It is often denoted to be the *inverse* operator to the halfedge collapse (Figure 44).

5.3.4 Edge-swap

Let P, R_1, Q and Q, P, R_2 be two adjacent triangles. Edge-swapping means replacing the common edge PQ by the edge R_1R_2, which leads to the triangles P, R_2, R_1 and Q, R_1, R_2.

Note that the edge-swap operator also can cause topological inconsistencies. The rule here is, that if the edge which results from the swap-operation already exists, the edge-swap would cause double edges and is therefore not allowed. Also since the valences decrease by one at each vertex, they should be at least four before the flipping operation.

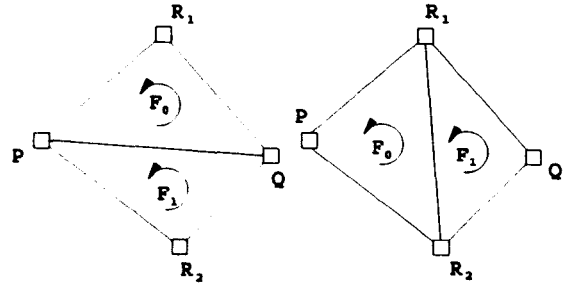


Figure 45: Edge-swap operation

Chapter 6

Polygonisation of Range Views

The discussion in Chapter 3 has shown, that each of the proposed polygonisation approaches has some specific problems. The $3D$ point cloud triangulation approaches are either not reliable, accurate or fast enough for our purpose. These approaches are therefore not considered, even if they are the most general approaches. Since we are interested in reconstructing the surface, defined by the data points, as accurate as possible, implicit surface triangulation also does not seem to be senseful. Therefore an approach, based on the algorithm of Karbacher [Kar97] has been developed. First one mesh for each view is generated. Since the range data to be processed is not always structured in matrices, a scanline based triangulation method has been developed. This triangulation covers corner detection and step discontinuity constraints. Then the meshes are limited such that they do not overlap each other. In contrast to the approach of Karbacher we introduce a quality criteria which guarantees, that triangles with bad quality are eliminated first. We also introduce an operator, which decides weather a triangle overlaps or not. This operator is reliable in any case, fast and simple to implement. After limiting, all meshes are combined using the *gap bridging* approach. We extend the method such that topological inconsistencies, which are eliminated in Karbacher [Kar97] afterwards, are detected and handled in advance. We are therefore able to reduce the number of holes, generated in this pre-processing step. Post-processing in order to eliminate geometrical and topological failures in the mesh is therefore not required. Remaining holes in the resulting meshes are closed using a $3D$ polygon triangulation method. Then, all data points are measured from the generated surface and reintegrated, if the distance is greater than a user defined

threshold. The reintegration is done using the face- or edgesplit operator. The meshes are constantly optimised during integration rather than once, afterwards. Figure 46 illustrates the principle of multiple view combination.

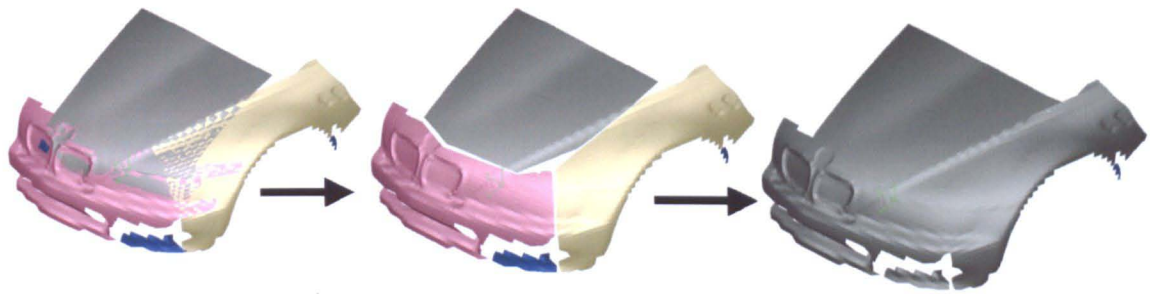


Figure 46: Principle of Multiple View Combination.

We also use a specific optimisation functional, which leads to smooth meshes. The method is closely related to the approaches in $2.5D$, proposed by Dyn et.al. [DLR90].

The present algorithm for polygonisation guarantees, that the distance of any data point from the reconstructed surface is less than a user defined tolerance. The surface is smooth and visually pleasing and very close towards the real surface of a part.

6.1 2.5D Triangulation

The 2.5D triangulation method developed in this work is restricted to data sets with given sensor orientation consisting of parallel and ordered polygons. This requirement is satisfied by most of the digitising systems available today, or data can be converted into such a description. These data sets are generated by tactile measuring machines, range images and data sets from CMM controlled range laser with fixed laser orientation. Holes in the data set are represented by passive segments in the polygon. Holes can either be generated and marked by the scanning system, or additionally they can be generated due to step discontinuity constraints. It is assumed, that the measured surface is locally continuous if the $3D$ Euclidean distance between adjacent measurements is less than a distance threshold, $DistMax$. Any two neighbouring polygon points with a distance greater than $DistMax$ are not connected, which leads to holes in the surface. A value of $DistMax = 4 * LineStep$ works well in practice. The *Linestep* parameter is the average point distance between neighbouring sample

data points.

The polygons can be filtered in advance to reduce the number of triangles. In practice it is recommended only to filter data points, if a very high number of triangles is expected without filtering. The quality of the meshes can be improved significantly if first a high number of triangles is generated and then the resulting mesh is filtered afterwards. Still, a maximum resolution is guaranteed by the polygon filtering approach (section 4.2.1), if a maximum resolution will not be exceeded.

A data set is triangulated by connecting points of two neighbouring polygons. Each polygon-point will get a vertex in the resulting mesh. The polygons are ordered in parallel. Triangles are given by connecting the closest vertices of the two neighbouring polygons, if their distance is less than the step discontinuity threshold $DistMax$. In addition, a pre-processing step detects sharp corners in the polygon. Corners are detected using the approach derived in section 4.2. When connecting the data points of different polygons, a preference for connecting two *corner vertices* is added. The two nearest corner vertices of different polygons are connected, if their distance is not greater than the threshold $DistMax$. This *corner preference* leads to a better reconstruction of corners in cases, where the scanlines of a laser or tactile system are perpendicular to the corner of the part. The corner detection is not very meaningful in the case of photogrammetric data. The predefined resolution does not allow precisely corner digitisation and a reconstruction is not useful at this stage.

A second preference is applied to the first and last vertex of a scanline segment. If the distance of the first point of a polygon P_i towards the first point of the polygon P_{I+1} is less than $DistMax$, both points are connected even if there are data points on P_{i+1} with smaller distance. The *endpoint preference* leads to a smoother boundary of the resulting mesh.

Figure 47 illustrates the principle of the triangulation process. The polygons 0 and 1 are triangulated first. In a) polygon end points are shown in light grey, corner vertices are shown in dark grey. In b) the areas to be triangulated are defined. In c) the resulting triangulation of the polygons 0 and 1 is shown. The process then continues with polygon 1 and 2 etc.

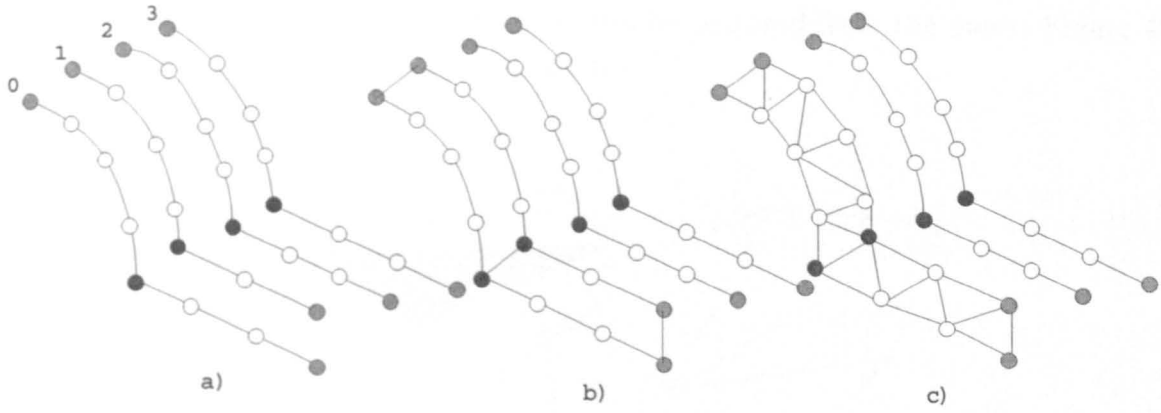


Figure 47: Triangulation of 2.5D range data.

Since every data point will get a vertex in the resulting mesh, it is guaranteed, that the mesh interpolates the given data points. The object is usually digitised with different sensor orientations and triangulation is applied to every data set.

A great advantage of the 2.5D triangulation process is the speed of the triangulation. Since the data is already sorted, it is a linear process and even for large data sets it only takes seconds to generate a triangulation. Especially for tactile and laser measured data sets, the corner detection generates high quality meshes with good corner representation. The step discontinuity reduces the amount of false edges in the resulting mesh to a minimum. Also the data points are interpolated, rather than approximated. The resulting mesh is therefore very near to the real surface.

6.2 Mesh Limiting

Meshes, generated from different range views have to be combined in order to generate one mesh covering all images. The first step in combining these meshes is to remove redundant triangles in overlapping regions. Let M_1 and M_2 be two overlapping meshes. Triangles of M_1 should be removed such that the two meshes do not overlap. Overlapping triangles can be determined using an offset approach. Therefore an operator *FacetOverlap* is defined. This operator decides, wheather a triangle overlaps another mesh or not. We therefore *thicken* mesh M_1, in particular any triangle T_i of mesh M_1, with a predefined offset $Offs$. Then, any *thickened* triangle is checked, wheather it intersects M_2 or not. If a triangle T_i intersects M_2, the *FacetOverlap*

operator returns FALSE, and the triangle can be removed from the mesh. Figure 48 illustrates the principle of the overlap routine.

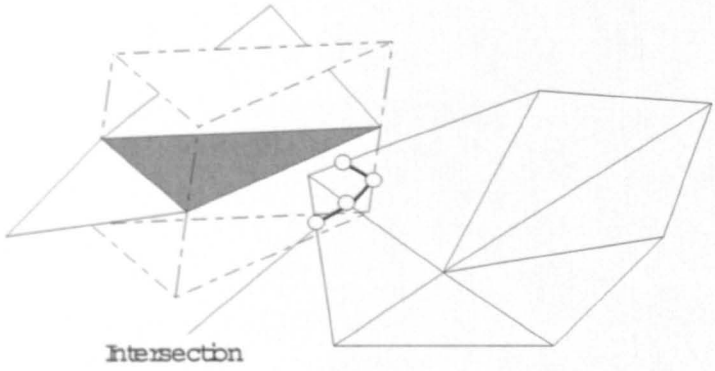


Figure 48: Determining overlapping using the offset criteria.

The threshold $Offs$ from which overlapping is determined can be chosen by the user. When choosing $Offs$, the accuracy of the scanning, the smallest *thickness* of the part as well as the resolution of the scanning system should be taken into account. In practice a value of $Offs = 2 * Linestep$ has been shown to be sufficient and reliable in most cases.

The combination of the two meshes is done in two steps. Every mesh consists of *good* and *bad* triangles. The idea is first to remove only overlapping triangles with a bad quality from $M1$. Then in the second step, triangles from $M2$ are removed such that the meshes do not overlap.

The triangle quality is determined using the scanner orientation. For an optical scanning system, the quality is derived from the angle between the scanner orientation (z-axis) and the triangle normal. A small angle is equal to high quality. For tactile measured data, the triangle quality is calculated from the angle between the triangle normal and the direction of line feed. A small angle means bad quality. It is necessary to define two different quality functions for tactile and optical systems, since tactile systems generate data with good resolution and quality in the scanline direction, even if the angle from the scan-axis is nearly 90^o.

In the first step of the algorithm, only triangles with a quality less than a threshold Q_{min} are processed. The $Qmin$ value is an angle value. In practice, a value of $Q_{min} = 45^o$ works well and leads to good results. If the quality of the triangle T_i is less than the threshold and if $FaceOverlap(T_i) = TRUE$, the triangle is removed

from the mesh. In the second stage, the quality criteria is dropped and all triangles are checked, if they are overlapping. This sequence guarantees that first redundant triangles with bad quality are removed and that afterwards only triangles with good quality overlap.

In order to allow sequential processing, we define a master mesh M_0 and a secondary mesh $M_i, i = 1, ..., N-1$. M_0 is the combined mesh, M_i is a mesh to be combined with M_0. Only triangles denoted to be of bad quality and overlapping M_i are removed from M_0. After that all triangles of M_i overlapping M_0 are removed from M_i. Finally all triangles of M_i are added to M_1 and the process is restarted with M_{i+1}. The data points of each processed range view can be deleted in order to save computing storage.

The limiting algorithm can therefore be formulated as follows:

```
I=0; while (I< Number of range views) {
  Triangulate range view I
  if (I>0) {
   Limit Mesh(0) towards Mesh(I) using the quality criteria
   Limit Mesh(I) towards Mesh(0) such that both meshes
     do not overlap
  }
  Add Mesh(I) to Mesh(0)
  Delete range view
  I++;
}
```

The limit process can be formulated for both quality and non quality mode in a common function *Limit* with the parameter $Mesh(0)$, $Mesh(1)$ and a boolean *QualityMode* as follows:

```
J=0; while (J<Number of triangles in Mesh(0)) {
  if (QualityMode==0 or TriangleQuality(T0(J))<Qmin) {
    if (TriangleOverlap(T0(J),Mesh(1)) {
      DeleteTriangle T0(J) from Mesh(0)
    }
```

```
    }
  }
```

Clearly the speed of the algorithm depends on the speed of the TriangleOverlap routine. For a fast and efficient implementation, first only the vertices in the mesh to be limited are checked wheather they overlap or not. Therefore the distance of any vertex towards the mesh is measured. If the distance of a vertex towards the mesh is less than $Offs$ and the plummeting point does not lie on the border of the mesh, the vertex is said to be *overlapping*. Any triangle in the fan around an overlapping vertex V can be removed. Removing all triangles in the fan around V sometimes leads to the removement of vertices in the orbit around V, too. If all triangles adjacent to this vertex are also removed, these vertices must not be measured which reduces the number of vertices to be checked.

If all triangles with one or more vertex overlapping are removed only partially overlapping triangles remain. There are two cases which are of interest. First, a triangle might overlap the other mesh fully. In terms of the offset approach this means, that mesh 1 does fully lie within the *thickened* triangle. This in contrast means that any vertex of mesh 1 does overlap mesh 2. These triangles are therefore removed in a subsequent step. Secondly a triangle might overlap only at its edges. Since these triangles overlap at the boundary of mesh 2, it only has to be determined wheather the triangles do overlap the boundary of mesh 2, and in particular boundary edges of mesh 2. A fast and efficient calculation of this intersection can be calculated as follows: First, the triangle T and mesh 2 are transformed into a co-ordinate system such that T does lie within the xy-plane of this co-ordinate system. Then the intersection of the three edges with the boundary of mesh 2 is calculated. Therefore first, triangles nearby the edges are determined. Then all open edges of the nearby triangles are checked wheather they intersect. If an intersection is found, the z-height of the intersection point is checked. If the absolute value of z is less than $Offs$, the edge and therefore the triangle overlaps and the process terminates. Figure 49 shows the different cases, where a triangle does overlap or not.

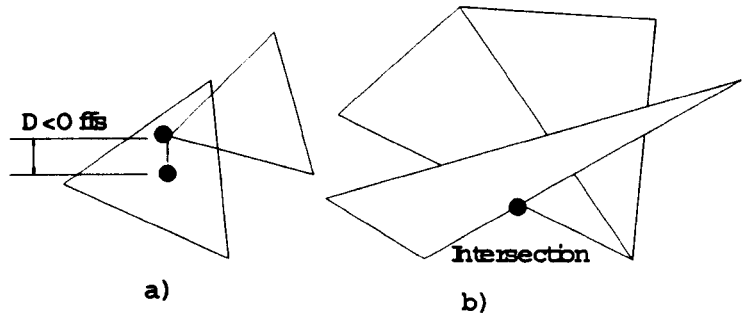


Figure 49: Vertex and triangle overlapping.

Finally not all the triangles in a mesh have to be checked wheather they overlap or not. Triangles far away from mesh 2 do not have to be measured. We therefore introduce a heuristic of how to determine wheather a triangle has to be checked or not. If the distance of any vertex of the triangles is greater than $Offs$ the triangle is not overlapping. This heuristic works well, if the triangles to be checked are almost of equal length, which is given in this case using the proposed $2.5D$ triangulation. This heuristic reduces the number of triangles to be checked to a minimum and therefore computing time can be saved.

6.3 Connecting Meshes with Gap Bridging

After the limitation process, the meshes do not overlap each other. Between any mesh, there are gaps, which have to be filled in order to generate a closed mesh. Karbacher [Kar97] introduced an approach, which is called *gap bridging*. Any edge on the border of one mesh is connected with boundary vertices on the other mesh. The vertex with the shortest distance from the edge is chosen for the connection. The presented approach is comparable to the algorithm derived within this work. The gaps between adjacent meshes are also closed with triangles, created by one boundary edge and one boundary vertex. Since all meshes are already combined into one single mesh M_0, the gaps within this single mesh are closed. All boundary edges of M_0 are possible candidates for gap bridging. In an initialisation stage, these boundary edges are checked as to see if they are connectable with a boundary vertex. Any vertex within a maximum distance $Dist_{Max}$ from the edge is a possible candidate for building gap triangles. Only those vertices are allowed, which lead to non overlapping triangles.

All possible candidates are stored within a priority queue. The priority is inversely proportional to the vertex distance towards the midpoint of the edge, $Prio = 1/Dist$. The idea behind this ordering is first to bridge small distances since short connections between vertices are more likely to describe the original surface. Also the chance that these triangles do not overlap other regions of the mesh is greater than with larger triangles. This is in contrast to the approach of Karbacher [Kar97], where an arbitrary sequence was chosen.

The triangle with the highest priority is taken first for building a gap triangle. After a gap triangle is introduced, one or two more boundary edges might have been generated. These edges might also be candidates and are introduced into the priority queue if they are connectable to the remaining boundary vertices. The process terminates if no more valid connections can be found.

The algorithm can therefore be defined as follows:

```
//-Initialisation-----------------------------------------
for all boundary edges in a mesh {
  Calculate all boundary vertices Vi within a distance<DistMax
  for all these vertices Vi {
    if VertexValid(Vi) {
      Save the vertex in list,
    }
  }
  if List is not empty {
    Sort the list, vertices with shortest distance first.
    Save the list as an attachment to the edge.
    Add edge E to priority queue, priority=1/shortest distance
  }
}
//-Triangle generation------------------------------------
while (Queue  is not empty) {
  Determine the edge E with highest priority and remove E
    from the queue
  Get first vertex V in the vertex list attached to E
  Create a triangle T, consisting of E and V
```

```
if (Not FaceOverlap(T,Mesh)) {
  Add Triangle to the mesh
  for all three edges Ej adjacent to T {
    if (Ej is boundary edge) {
      Check Ej analogue to the initialization stage
    }
  }
} else {
  delete V from the vertex list.
  if another vertex V+1 is in the vertex list
    Get this next vertex V+1 from the vertex list
    Reintroduce E into the queue with the priority of
      this vertex.
  }
 }
}
```

The $VertexValid$ operator reduces the number of possible candidates for the gap bridging process. There are some simple and fast criteria wheather a vertex is a possible valid candidate or not. First, vertices with a distance from the midpoint of the edge larger than a threshold $DistMax$ are not valid. The beginning and end vertex of the edge is not allowed, either. If the plummeting point of a vertex V on the edge E does lie on the boundary of E, the vertex is also not allowed. This guarantees that triangles consisting of neighbouring vertices on an almost linear boundary are not generated. A third criterion measures the angles between the possible triangle and adjacent surface normals. First, three surface normal vectors $Nv(i), i = 0, ..., 2$ at the three vertices of the new triangle are calculated. This is done by calculating the average of all triangle normals in the fan around these vertices. Then, the dihedral angles between the triangle normal and the three vertex normals at adjacent vertices are calculated. If one of these three dihedral angles is greater than a threshold $AngleMax$, the triangle normal does not project into the direction of the surrounding surface, and the triangle is not allowed to be generated. Choosing the threshold $AngleMax = 90^o$ guarantees, that wrongly oriented triangles are avoided. Finally it is determined, wheather the triangle overlaps other vertices. Therefore

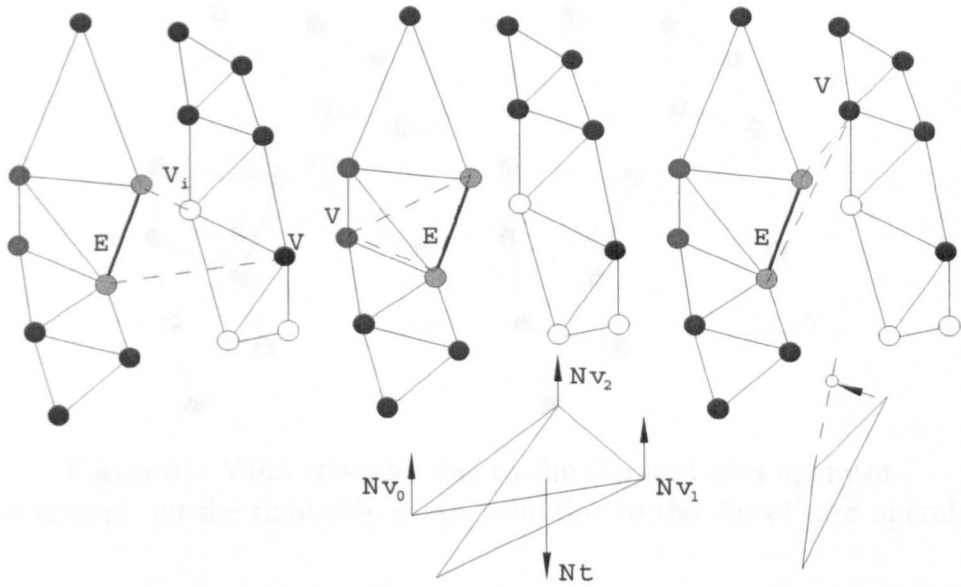


Figure 50: Illegal gap-bridging candidates, detected with the $VertexValid$ operator.

the distance of any vertex towards the triangle T is measured. If the distance is less than $Offs$ and the plummeting point is not on the boundary of T, the triangle overlaps this vertex and is therefore not valid. Now only vertices remain as possible candidates for building a triangle, which do not overlap any other vertex, which do not generate wrong oriented triangles, and which do not connect coplanar vertices. Figure 50 illustrates the principle of the $VertexValid$ operator. Here all vertices are within the distance $DistMax$ towards the edge and are therefore possible candidates. On the left side, the triangle is not allowed due to the overlap criteria. In the middle, the triangle is determined to be not valid due to the angle criteria. On the right side, the triangle is not valid due to the boundary criteria.

Figure 51 shows examples of possible triangles to be added to the mesh. Note that the triangle on the right side is not valid but could not be detected with the $VertexValid$ operator. To avoid the generation of such triangles, a $FaceOverlap$ operator analogue to the limitation process is introduced. Again, it is only necessary to examine wheather these triangles do overlap boundary edges. The approach using a recursive box structure over triangles, used for the limiting approach is not useful since the number of triangles dynamically changes with each new triangle. Also only the triangles or edges at the boundary of the mesh are of interest. Since the number of vertices does not change during the gap-bridging process, the edges near

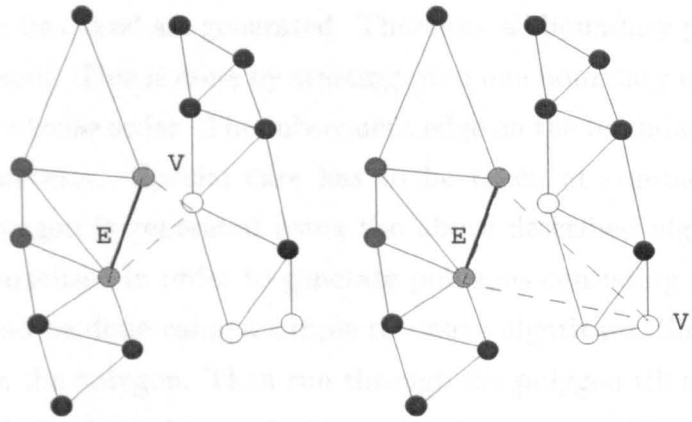


Figure 51: Valid triangles due to the $VertexValid$ operator.
The triangle on the right side is not valid due to the $FaceValid$ operator.

to the triangle to be checked are determined using the vertices at the boundary of the mesh. First, a binary search tree over all the vertices at the boundary of the mesh is introduced. This allows quick determination of all vertices near to the triangle. Any vertex with a distance less than a threshold $DistMax$ towards one of the three triangle vertices is checked. Then for all these vertices, adjacent boundary edges are determined and double edges are eliminated. Finally, analogous to the approach in the limiting stage, wheather the triangle intersects one of these edges in the triangle plane, and wheather the intersection points have a 3D distance less than $Offs$ is checked. The $FaceValid$ operator returns FALSE, if a triangle with intersection was found.

6.4 Hole Filling

In the above describtion of the gap-bridging approach, there are cases where holes remain in the resulting surface. Some of these holes remain because of the gap-bridging criteria, which are quite restrictive in order to guarantee an error free gap bridging process. Holes typically remain in areas of high curvature at convex corners with an angle smaller than 90^o. Many of these holes can be closed. Since at this stage of the mesh generation process holes in the mesh can be determined and classified, the hole filling can be done locally with less restrictive criteria.

Hole filling is done using a simple polygonal triangulation approach in $3D$. First,

the polygons P_i to be closed are generated. Therefore all boundary polygons of a mesh have to be calculated. This is done by starting from one boundary edge and collecting next edges in a clockwise order. The subsequent edge on the boundary can be obtained using the *star* operator. Special care has to be taken at complex vertices. First any boundary polygon is generated using the above described algorithm. Then the polygons are decartelised in order to generate polygons consisting of unique vertices. Decartelisation can be done using a simple recursive algorithm. Simply find the next complex vertex in the polygon. Then run through the polygon till the complex vertex is found again. Split the polygon at this vertex into two and restart the algorithm with these splitted polygons, until no complex vertex remains in a polygon.

Now any polygon with an area $A(P_i)$ less than a threshold $Area_{Min}$ is a candidate for hole filling. The polygon area is calculated by projecting the polygon in the xy, xz and yz plane of an arbitrary co-ordinate system. $A(P_i)$ is equal to the maximum of the three surrounding boxes in these planes. A value of $Area_{Min} = 4 * Linestep$ works well in practice.

The triangulation of the polygons is done using a simple *greedy* algorithm. Any three adjacent vertices in the polygon are candidates for building a triangle. In $2D$, a triangle is not allowed to be generated, if the corner is not convex or if the triangle overlaps other vertices. For a $3D$ polygon, convexity is not defined. Therefore the dihedral angles between adjacent triangles are taken into account. These are the new triangle candidates and their neighbours on the original mesh. If the dihedral angle between the normal of a new triangle and the vertex normals at its three vertices is larger than a threshold Ang_{Max}, the triangle is not valid. A value of $Ang_{Max} = 120°$ works well in practice. Also we measure overlapping by projecting all vertices in the polygon except the three, the triangle consists of, onto the triangle. If a plummeting point does lie within the triangle and the distance is less than an offset $Offs$, it is not valid. The $Offs$ value is the same as in the gap bridging process. Figure 52 illustrates the principle of this *FaceValid* operator. In b) the triangles are shown which are not valid with respect to the above described operator. In c) the valid triangles are shown.

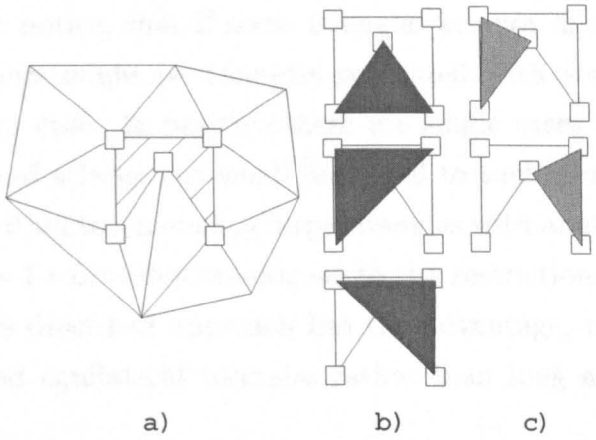


Figure 52: The *Facet Valid* operator prevents the hole filling algorithm from generating overlapping triangles.

For all valid triangles, the one with the best quality is taken and generated. Therefore possible candidates are stored in a priority queue and the one with highest priority is generated and removed from the list.

The triangle with the smallest surrounding circle is the one with highest priority. After a triangle has been generated, two edges are removed from the polygon and one new triangle is generated (except for the last triangle, where no edge remains). Three edges, the new one and its previous and successive edge on the boundary are recalculated and updated in the priority queue. The algorithm terminates, if either no triangle remains in the queue, or if no more valid triangle could be found. Figure 53 illustrates the principle of the hole filling algorithm.

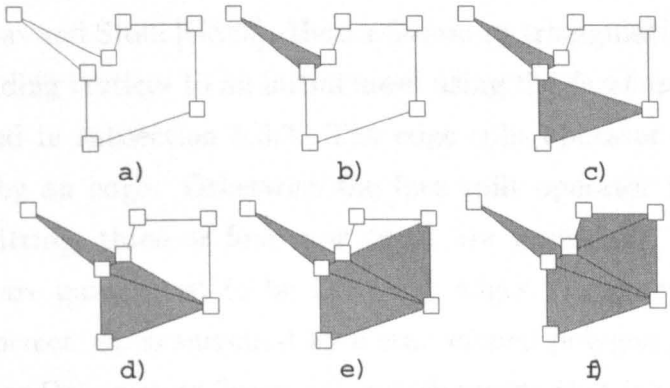


Figure 53: Principle of the hole filling algorithm.

It is important to notice, that if some triangles are not allowed to be generated in this algorithm, there might be triangles generated with very low quality which is not desirable in our case. In practice there are single cases in which overlapping triangles on one side of a large but small hole lead to such problems. Therefore we also restrict the algorithm to generating large triangles with an maximum edge-length of $EdgeLength_{Max} = 4 * Linestep$, analogous to the restrictions in the triangulation approach. The above described approach has the advantage, that due to the priority criteria, small and equilateral triangles rather than long and thin triangles are preferred.

6.5 Mesh Refinement using Vertex Insertion

As stated earlier, it is desired to guarantee that any data point lies within a user defined tolerance of the mesh. The meshes generated by now do not guarantee this requirement. Measuring the distance of the removed vertices from the mesh shows, that there are data points left whose distances are larger than the user defined threshold. This is always possible since due to the limiting approach not all the vertices are taken into account. In doubly curved areas none of the data sets might be able to cover the geometry nicely, but the combination of them does lead to a nice description. Combining the data sets of all images into one mesh leads to a denser distribution of data points and to a more accurate representation of the part.

The combination of two data sets is done using vertex insertion and reorganisation using edge-swaps. The idea is derived from the vertex insertion approach in $2D$, presented by Guibas and Stolfi [GS85]. Here a *Delaunay* triangulation can be obtained by sequentially adding vertices to an initial mesh using the *facet split* or the *edge split* operator, presented in subsection 5.3.3. The edge split operator is used if the new vertex V_n is nearby an edge. Otherwise the face split operator is used to insert a vertex. After splitting, three or four new edges are generated. It can be proved, that these edges are guaranteed to be Delaunay edges. However, some of the old edges may be incorrect. V_n is surround by a star shaped polygon. The edges on this boundary are either Delaunay or "suspect", which means that it is not known if they pass the Delaunay (incircle) test. The incremental algorithm proceeds by choosing a suspect edge and applying the incircle test to it. It can be proved, that if the edge

passes the incircle test it is guaranteed to be a Delaunay edge and need not to be considered further. If the test fails, the edge is swapped using the *edge swap* operator. In that case, the new diagonal becomes Delaunay while the two sides opposite V_n become suspect. The algorithm terminates, when no suspect edges remain. Figure 54 illustrates the principle of the incremental Delaunay algorithm. Since the meshes to be processed are 3 dimensional rather than 2 dimensional, the incircle test is replaced by comparing the minimum of the inner angles of the two adjacent triangles of an edge before and after the edge swap operation. If the minimum angle can be maximised, the swap operator is applied. This is the $3D$ analogue to the Delaunay criteria.

Now, any vertex, whose distance is larger than the user defined threshold D_{max} is a possible candidate for vertex insertion. Data points are only inserted into the mesh in the direction of the sensor orientation. This is done to avoid overlapping triangles, which could be generated if more than one vertex has to be inserted into one facet. After a triangle or edge has been splitted, the mesh is locally optimised using the above described algorithm.

Since during the optimisation procedure, some of the vertices not considered in the mesh generation process but within the user defined threshold, might be outside the distance threshold after edge swapping. These vertices have to be evaluated again. Since not all the vertices have to be remeasured, it is only necessary to recalculate those, whose plummeting point was on a facet, which has been changed in a subsequent step. For an efficient implementation, we store a pointer or index of a data point in a list related to the facet, in which the plummeting point of the data point lies. If the facet geometry changes in a subsequent optimisation step, the data points related to this facet have to be remeasured towards either the three or four triangles, generated by the split operation or the two, changed by the edge swap.

After refinement it is guaranteed, that any data point has been measured from the generated mesh. Any vertex with a distance greater than the threshold from the generated surface has been integrated. The generated mesh therefore guarantees to be within a user defined approximation tolerance of the data points. Figure 54 illustrates the principle of mesh refinement using the incremental Delaunay algorithm.

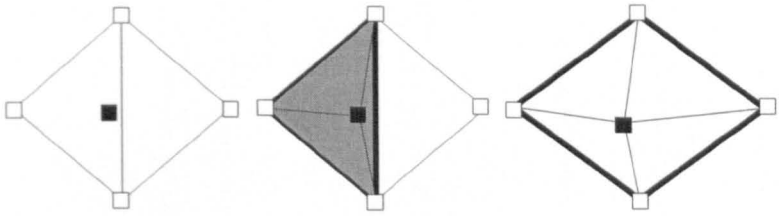


Figure 54: Principle of the incremental Delaunay algorithm.

Chapter 7

Mesh Optimisation

An important task in processing triangular meshes is optimisation. The tasks to be solved are derived from the requirement specification. The most important two algorithms are

- Noise filtering and
- Triangle decimation (or vertex filtering).

Noise filtering in meshes is necessary to reduce the noise, produced by the scanning system or from surfaces with a high roughness such as styrofoam or equivalent. Also at sharp corners or small radii, data from laser or photogrammetric systems tend to be noisy because of diffuse reflection. Noise can result from the vibration of the measuring system, the wavelength of the laser, the surface finish of the real part or other problems.

The number of triangles heavily influences the handling of the meshes and the speed of pre-processing. Triangulation tools based on interpolation easily produce millions of triangles and more for a shape, which can be described precisely enough with a much smaller number of triangles. Even if todays fastes graphics hardware can handle those high numbers of triangles, it is not easy to handle such meshes and to design fast algorithms. It is much easier to find artefacts, holes or other irregularities in a mesh with a less number of triangles compared with meshes with more triangles describing the same topology. Also with special mesh reduction techniques, an optimisation in terms of triangle geometry, smoothness and performance can be achieved.

It is important, that both algorithm produce equilateral (*round*) triangles rather than long thin triangles. The roundness of the triangles is important for many applications. Especially in finite element analysis it is important to work with triangles with good aspect ratios. The rounder the triangles are, the better the result of the computation is. In fact, most of the FEM-Systems are not able to handle triangles with an aspect ratio less than 0.01. Also for calculation of normal vectors and curvature at vertices, which is important for applications such as smoothing, or harmonic embedding of meshes (important for parametrising meshes [Kob98b]), it is important to work with round triangles. Finally it is also important for psychological reasons since a mesh with round triangles looks better and more accurate than a mesh with long thin or small triangles, even if they describe the same geometry.

The approach for triangle decimation, implemented within this work, is based on the algorithm proposed by Kobbelt et.al.([CKS98b], [CKS98a]). We present an algorithm, which reduces the number of vertices (and therefore the number of triangles) in a mesh while preserving accuracy of the mesh with respect to the data points, smoothness and triangle roundness. Three different approaches have been observed for noise filtering. A discrete fairing approach, based on the work of Kobbelt [Kob97] is presented, which is based on curvature minimisation. The *LAPLACE* smoothing approach, proposed by Nowottny [Now99] is examined and finally a smoothing approach by reorganising edges is introduced.

7.1 Triangle Decimation

The approach, implemented within this work, is based on the algorithm proposed by Kobbelt et.al.([CKS98b], [CKS98a]).

The idea is to reduce the number of vertices in a mesh by sequentially applying half-edge collapses to a mesh. The halfedge collapse operator has the advantage, that no further geometry has to be introduced to the mesh, that the operator is very simple and fast and that the operator is reversible, which leads to progressive meshes. The aim of the decimation algorithm is to reduce the number of vertices in a mesh while preserving the accuracy and quality of the surface. The accuracy and quality of a reduced mesh is defined in terms of the approximation error, triangle roundness and tangential continuity between adjacent triangles. Clearly a reduced surface should

approximate the original surface within a user defined threshold. It is also desired from the requirement specification, to generate round triangles rather than long and thin triangles. Finally, the smoothness of the surface should be preserved or optimised. Measuring only the approximation error would lead to large discontinuities in tangential continuity in areas of higher curvature. Figure 55 shows an example of a decimated polygon. On the left side, the polygon was filtered ignoring the tangential continuity between different polygon segments. On the right side, the polygon was filtered while taken into account the tangent vector between different polygon segments.

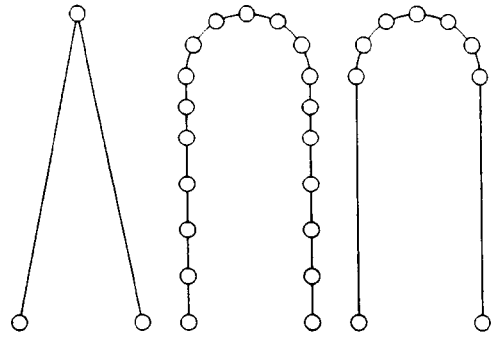


Figure 55: Vertex removement with and without angle criteria.

For a further description of the algorithm, two operators, the *CollapseValid* and the *CollapsePriority*, are defined. The first operator *CollapseValid* is a binary operator and decides wheather a halfedge collapse is allowed in terms of topological correctness and geometrical accuracy. In subsection 5.3.2 it is shown, that a halfedge collapse can cause topological inconsistencies in some cases. If the collapse causes topological errors, the operator is not applied. The *CollapseValid* operator will then return FALSE. The approximation error of the reduced mesh with respect to the data points is calculated using a distance measurement. Therefore, a collapse operator is applied *virtually*. Then the local distance of the removed vertex from the resulting mesh after the collapse is measured. Since the operation is done locally, only the distance from the triangles, changed by the operator has to be measured [CKS98b]. A fast distance function using a *slot distance* approach is described in [CKS98b]. If the distance is larger than the user defined threshold $Dist_{Max}$, removal of this vertex is also not allowed. If the vertex is allowed to be removed, it is not deleted from memory but stored within a list, related to the triangle with the shortest distance. If

the triangle geometry changes in subsequent collapses, the vertex again is measured towards the resulting surface. Any vertex is therefore measured in each decimation step. It can then be guaranteed, that any vertex does lie within the user defined threshold from the reduced surface. The decimated surface therefore approximates the vertices of the original mesh within the threshold.

Figure 56 shows the principle of this distance measurement. Vertices to be removed are shown in black. After the collapse, the distance towards the locally changed triangles, shown in light and medium grey is measured. The triangle in medium grey is the one with the shortest distance to the vertex. In the second step, the first and second removed vertex is measured towards the triangles. Again the medium grey triangle is the one with the shortest distance to both vertices. They are both stored within this triangle. Note, that in the final operation the collapse would cause overlapping triangles. These triangles are shown in dark grey.

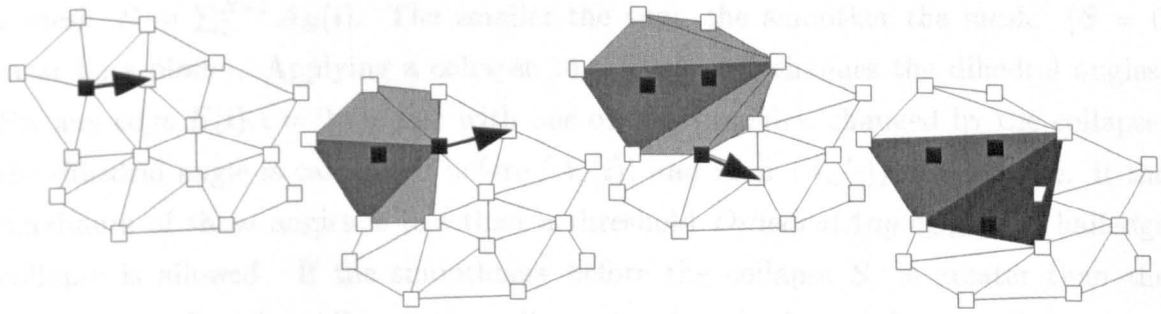


Figure 56: Vertex removement with halfedge collapses.

The roundness of a triangle is defined by the *aspect ratio*. The aspect ratio R measures the roundness of a triangle and is calculated with $R = H/L$, where H is the height of the triangle and L is the length of the longest edge. The aspect ratios of all N triangles, changed by the collapse, are calculated before $R_v(i)$ and after $Rn(i)$ the collapse. If the minimum aspect ratio $Min_{i=0}^{N-1}(R_n(i))$ is greater than the threshold $AspectRatio_{Min}$, or if $Min_{i=0}^{N-1}(R_n(i)) > Min_{i=0}^{N-1}(R_v(i))$, the collapse is allowed. In practice a value of $Ratio_{Min} = 0.1$ has been discovered to be best and well suited for all applications. With $Min_{i=0}^{N-1}(R_n(i)) > Min_{i=0}^{N-1}(R_v(i))$ a collapse is allowed, if the minimum roundness is maximised. This is necessary and desirable. If there are triangles adjacent to the vertex to be removed with a bad *aspect ratio*, the collapse might improve triangle roundness even if there are still triangles with an aspect ratio

less than the threshold.

Smoothness is measured using the dihedral angles at closed edges between the surface normals of adjacent triangles $A(i) = \angle(N_{v1}(i), N_{v2}(i)), i = 0, ..., N$ (Figure 57).

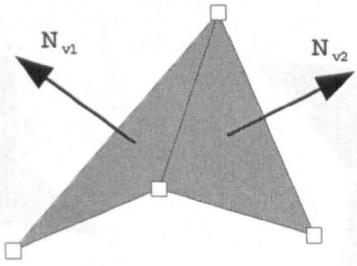


Figure 57: Dihedral angle between adjacent triangles.

The smoothness of a mesh can be calculated by the sum of all dihedral angles in a mesh, $S = \sum_0^{N-1} A_E(i)$. The smaller the sum, the smoother the mesh. ($S = 0$ must be a plane). Applying a collapse to a mesh now changes the dihedral angles. For any edge $E(i), i = 0, ..., Enr$ with one or two triangles, changed by the collapse, the dihedral angle is calculated before ($A_v(i)$) and after ($A_n(i)$) the collapse. If the maximum of these angles is less than a threshold $DihedralAngle_{Max}$, the halfedge collapse is allowed. If the smoothness before the collapse S_v is greater than the smoothness after the collapse, the collapse is allowed, also. Otherwise the collapse is not allowed and the *CollapseValid* operator returns FALSE. Again, the functional allows the mesh to be improved by the collapse in terms of smoothness, even if the threshold of $DihedralAngle_{Max}$ is not reached after a collapse. Restricting the $DihedralAngle_{Max}$ to be a value between 0^o and 90° preserves the mesh decimation from overlapping triangles. A value of $DihedralAngle_{Max} = 5^\circ$ is sufficient for all applications.

Figure 58 illustrates the local change of geometry after performing an edge collapse. The triangles, shown in dark grey are the triangles to be removed from the mesh. Medium grey are the triangles, whose geometry changes after the collapse. For these medium grey triangles, the roundness has to be computed. They are exactly the triangles in the fan around P before the collapse, except the two removed by the collapse. Triangles in medium and light grey are important for measuring the smoothness in a mesh. At all edges in the orbit of P, and at the edges which have

been changed by the collapse, the dihedral angle changes. Using these angles, the smoothness can be computed.

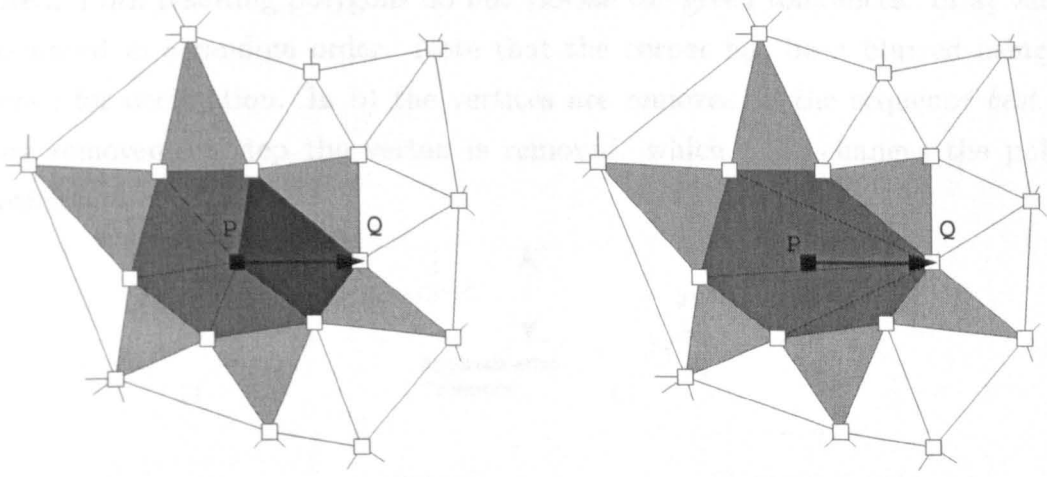


Figure 58: Local change of geometry after performing the collapse operator.

Finally a maximum edge length $edgelength_{Max}$ is introduced as a quality criterion. Since better results can be achieved in some applications with smaller triangles, removement is restricted, if the edges changed by the collapse are longer than a user defined threshold $edgelength_{Max}$.

To summarise, there are 5 criteria deciding, wheather a halfedge collapse is allowed or not

- Topological correctness.
- Approximation error in terms of distance of the removed vertices towards the decimated mesh.
- Triangle roundness must improve or be larger than a minimum roundness.
- The surface smoothness in terms of dihedral angles between adjacent triangles must improve or be larger than a threshold $DihedralAngle_{Max}$.
- The maximum edge length should not be larger than a user defined threshold $edgelength_{Max}$.

The *CollapsePriority* operator returns a double value Q which characterises the quality of a valid collapse operation. The idea is to apply the collapse operations

in the sequence *best first*. As illustrated in figure 59, the quality of a decimated surface (or here a polygon) heavily depends on the sequence, in which vertices are removed. Both resulting polygons do not violate the given tolerances. In a) vertices are removed in a random order. Note that the corner has been blurred using this sequence for decimation. In b) the vertices are removed in the sequence *best first*. In any removement step the vertex is removed, which least changes the polygon smoothness.

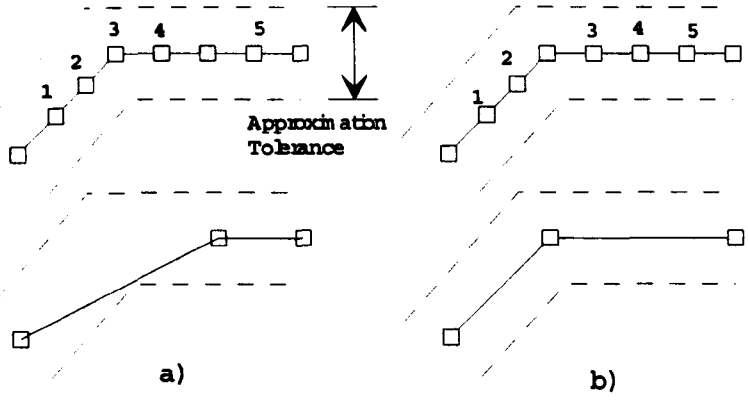


Figure 59: Different strategies produce different results in the decimation process.

The quality Q of a halfedge collapse is calculated from the smoothness criteria. This definition results from many tests with different criteria and from the fact, that the smoother the mesh is, the better the results are in subsequent applications. The smoothness is inverse proportional to the maximum dihedral angle at modified edges after the collapse $Q = 1/A_{max}$.

With these two operators *CollapseValid* and *CollapsePriority*, an algorithm can be defined for triangle decimation. The algorithm is divided up into the steps *initialisation* and *decimation*. In the *initialisation* step, any possible collapse is evaluated without applying the halfedge collapse directly. If the halfedge collapse is valid, then an index for this operation is stored in a priority queue. A priority queue is a sorted list of entries. The ordering of these entries is given by a priority which in our case is calculated with the *CollapsePriority* operator. A description and code for a priority queue can be found in the book *Algorithm* from Robert Sedgewick [Sed92]. The initialisation algorithm can be formulated as follows:

```
//-Initialisation----------------------------------------------
```

```
for all halfedges H in a mesh {
 if (CollapseValid(H)) {
  Insert H index into Priority Queue using CollapsePriority(H)
 }
}
```

Kobbelt proposes an optimisation for this initialisation. Rather than handling the halfedges, the vertices to be removed are evaluated. For all vertices, the halfedges around this vertex P are evaluated and the best halfedge is associated with this vertex. It can be shown [CKS98b], that this is sufficient to guarantee a *best first* sequence of decimation. The geometry of all edges adjacent to P changes after the collapse. It is therefore not necessary to store all halfedges since they have to be recomputed after the best collapse. Since the number of halfedges is much larger than the number of vertices, the storage required for the priority queue can be reduced. Also since the evaluation for all halfedges H_i adjacent to the vertex V is done one after another, the number of calculations for values such as triangle roundness and triangle normals can be reduced, if all locally required values are stored temporarily for this optimisation. The algorithm for initialisation can then be modified as follows:

```
//-Initialisation-----------------------------------------
for all vertices V in a mesh {
 PrioMax=0; Found=FALSE;
 for all halfedges H around V (calculated using the Star operator)
  if (CollapseValid(H) && CollapsePriority(H)>PrioMax ) {
   Save maximum values HMax=H; PrioMax=CollapsePriority(H);
   Found=TRUE;
  }
 }
 if (Found) {
  Save Hmax in a list related to the vertex V
  Insert V into the Priority Queue using the PrioMax
 }
}
```

In the second algorithm, the halfedge collapse operator is applied in the *best first* sequence to the mesh. After applying one collapse, all possible halfedge collapses, changed in validity or priority have to be recomputed. For a collapse from vertex P to vertex Q, clearly the validity and priority for the vertex Q has to be recomputed. Also all vertices V_i in the orbit around the vertex P have to be recomputed since the geometry of the triangles adjacent to P has changed. Finally any vertex in the second orbit around P has to be recomputed since for any possible halfedge collapse in this orbit the smoothness changes. Again a heuristic, described in [CKS98b] can be taken in order to reduce the amount of updates and therefore the number halfedge collapse evaluations. It is assumed, that the halfedge collapse with the highest priority around a vertex on the second orbit remains to be the one with the highest priority after the collapse. This is exact in most cases and works well in practice. For any vertex $V2_i$ in the second orbit around P, the only halfedge to be recomputed is the maximum halfedge, calculated in former steps.

The decimation algorithm can therefore be formulated as follows:

```
//-Decimation-----------------------------------------------
while (Queue is not empty) {
 //-Apply collapse-------------------------------------------
 Get vertex index with highest priority and delete this entry
 Get related halfedge collapse with highest priority from the
   vertex related list
 Apply Collapse(H)
 //-Queue update---------------------------------------------
 for all vertices in the orbit around P {
  Calculate CollapseValid and CollapsePriority analogue to
    initialisation
  Update Queue
 }
 for all vertices in the second orbit around P {
  Calculate CollapseValid, CollapsePriority only to the
    halfedge with maximum priority.
  Update Queue
 }
```

}

Note, that since the number of updates has been reduced such that only geometrical changes rather than the topological consistency has been considered, it is important to verify the topological consistency always before applying a halfedge collapse. It is also important to notice, that due to geometrical and topological changes after a collapse, there might be vertices removable which were not in former steps. These vertices are reintroduced within the update step and considered in subsequent removal steps.

Results and examples for triangle decimation are given in Chapter 8.

7.2 Fairing

Three different algorithms have been observed and developed for this work. A discrete fairing approach to reduce noise in a mesh while preserving the shape is suggested by Kobbelt [Kob97]. A mesh M is smoothed by moving the vertices V of a mesh such that the global energy of a mesh is minimised. The energy functional $\varepsilon[M]$ is defined by the weighted sum of the square curvatures κ_1 and κ_2 at each vertex.

$$\int_\Omega \kappa_1^2 + \kappa_2^2$$

or as a discrete function

$$\varepsilon[M] = \sum_i \omega_i \left(\sum_j \gamma_{i,j} V_j \right)^2$$

where $\gamma_{i,j}$ are the partial derivatives of the surface. Minimising therefore means solving the following equations:

$$\frac{\partial}{\partial V_i} \varepsilon[P] = 0$$

The second order derivatives for the curvature are approximated using local parameterisation and surface approximation. The sparse matrices of the equations are solved using Gauss-Seidel method. A more detailed describtion on this approach is given in the work of Kobbelt [Kob97]. The approach leads to noise reduction; high frequencies are filtered and the resulting surfaces are smooth and of high quality.

The above approach has one disadvantage. The quality of the smoothed surface depends on the quality of the curvature estimation at the vertices. For highly noisy

meshes and for meshes which large differences between the valences at vertices, the curvature estimation fails. Artefacts such as single spikes might be generated in this process. For these special cases, where highly noisy meshes are to be processed, a second approach has been implemented, which is based on the approach of Nowottny [Now99] and Karbacher [Kar97]. Nowottny suggests *LAPLACE* smoothing in order to improve the roundness of the triangles. A vertex V is moved into the centre of all vertices in the orbit around V as shown in Figure 60. The centering operator is applied sequentially to any vertex in a mesh. The analogue to this operator in terms of signal processing is the finite impulse response filter (FIR filter) which is used in order to remove high frequencies in a signal ([Blab]).

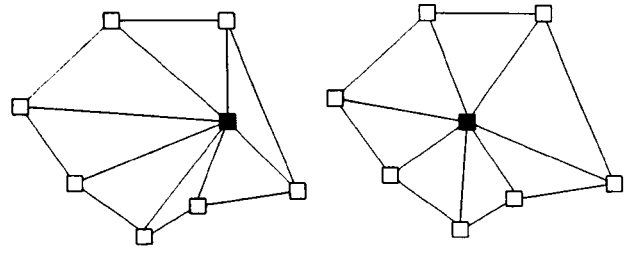


Figure 60: Vertex centering.

Applying the umbrella operator has the effect, that the noise in a mesh is reduced and that the mesh is straightened. The last property is very nice on one hand since the operation then leads to round triangles with a good aspect ratio. On the other hand there are two disadvantages using this operator. Consider an object such as a sphere. If a vertex is centred within its orbit, the resulting fan is flat and therefore does not lie on the objects surface. Applying the umbrella operator once or many times has the effect that the object *shrinks* and curved areas will become flat.

The second disadvantage appears when data with less noise but with corners is filtered. Corners will be blurred if the umbrella operator is applied to vertices at these corners. Therefore these vertices have to be excluded from filtering. Analogous to the approach, proposed for scanlines, a distance threshold is used in order to reduce the blurring effects. If the distance of the centred vertex V from the original fan around V is less than a user threshold, or if the distance of the original vertex from the fan around V after smoothing is larger than the threshold, the umbrella operator is not applied. Since at corners the vertex is usually moved further than on surface

areas with lower curvature, these vertices are not smoothed and therefore the corners are not blurred. The approach is especially suited for applications, where triangle roundness is more important than accuracy.

The last approach reorganises the topology of a mesh by swapping edges rather than changing the geometry by moving vertices. Most of the known approaches for triangulating range images do not consider smoothness criteria in the triangulation process. The approaches, described in Karacher [Kar97] and Hilton [Hil95] triangulate range images by simply connecting adjacent vertices in the 2D image plane. In 2D often the Delaunay triangulation is used in order to generate a triangulation. As shown in [DLR90], much better triangulation can be achieved when taking into account the 3D information rather than only the 2D positions. It is also shown, that any triangulation can be transformed into another triangulation using the edge swap operator and certain criteria, which decide wheather the swap should be applied or not.

The idea is now to transfer the results from the 2D approach to 3D triangular meshes. A swap operator is applied to an edge if the smoothness of the mesh can be improved. The smoothness is measured using the dihedral angles between adjacent triangles. Global smoothness is defined by the sum over all dihedral angles in a mesh

$$S = \sum_{i=0}^{Enr-1} Ang(E_i)$$

where Enr is the number of closed edges in a mesh and $Ang(E_i)$ is the dihedral angle at an edge E_i. The smoothness is measured before and after a possible collapse. If the value of S_v is greater than the smoothness value S_n after the collapse, then the operator is applied to the mesh. The swap operator only changes the geometry of an object locally. Since only the triangles, adjacent to the edge to be swapped are changed in the geometry, it is only necessary to calculate the dihedral angles at edges adjacent to this two triangles and at the edge to be swapped. The operator is applied in the sequence *best first* to the edges of a mesh. Similar to the triangle decimation approach every possible edge swap is first qualified. A swap operation is allowed if topological consistency is guaranteed after the edge swap and if the smoothness of the mesh is better after the operation. To avoid long and thin triangles, we restrict the swap operation in terms of roundness. If the minimum aspect ratio of the triangles adjacent to the edge to be swapped after the edge swap is less than before and if

this minimum is smaller than a threshold $AspectRatio_{Min}$, the operation is not valid. A value of $AspectRatio_{Min} = 0.1$ leads to good results in practice. The greater the difference between the smoothness before and after the edge swap, the higher the quality of the operation and the greater the priority is. Applying the operator in an ordered sequence rather than in an arbitrary sequence leads to much better results since the most discontinuous edges are swapped first. After each edge swap operation, the edges in the neighbourhood of E have to be recalculated. These are the four edges adjacent to the triangles F_1, F_2, changed by the swap operator. Updating these edges is sufficient to guarantee, that any geometrical changes have been considered. In order to consider the topological aspects, more updates are necessary. To reduce the number of updates after an edge swap, this topological checks are done just before the operator is applied rather than with the SwapValid operator.

The algorithm for edge swap smoothing can therefore be formulated as follows:

```
//-Initialisation-----------------------------------------------
for all edges E in a mesh
 if SwapValid(E)
  Prio=SwapPriority(E)
  Insert E with Priority Prio into a priority queue.
 }
}
//-Edge swap----------------------------------------------------
while (Queue not empty) {
 Get entry E with highest priority and remove entry from queue
 if SwapConsistency(E)
  EdgeSwap(E)
  for all edges E(i) adjacent to the triangles, adjacent to E {
   if SwapValid(E(i)) {
    if E(i) is already in the queue {
     Update Priority of E(i)
    } else {
    Insert E(i) with SwapPriority(E(i)) into the queue.
   }
  }
```

```
  }
}
```

Figure 61 illustrates the principle of the algorithm. The initial triangulation, shown in a), is an arbitrary triangulation of the object. In b) the edge with the highest priority is swapped. Note, that this is an edge at the corner of the object. In c) the final mesh is shown after optimisation.

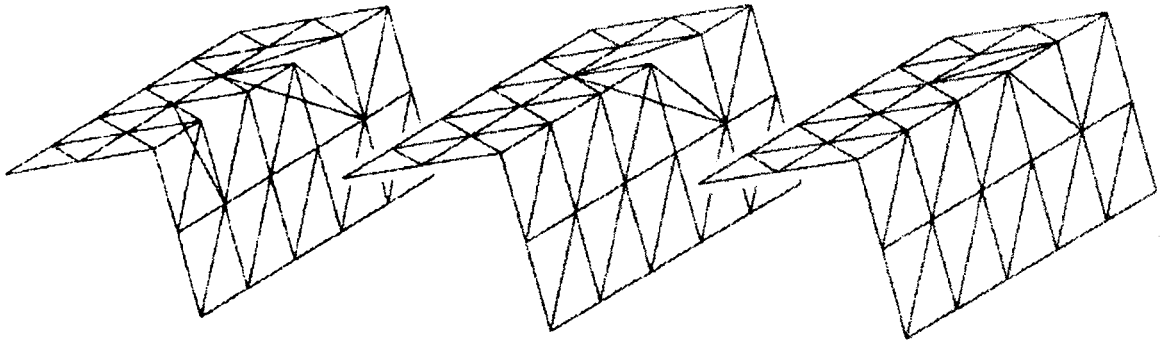


Figure 61: Optimisation with edge swapping.

Since the swap operator does not change vertex positions, the smoothness in a mesh is only optimised by retriangulating these vertices. This is a very nice property since it is guaranteed, that the positions of measured points are not changed. Since the sum of all the dihedral angles, changed due to the swap operation, are considered in the measurement, the operator especially works well at corners and small radii. This is in contrast to the approach of Choi et.al. [CSYL88], where the edge to be swapped is not taken into account in the smoothness measurement. The edge swap optimisation is applied to any mesh, generated with the polygonisation algorithm described in Chapter 6. It can also be applied to any mesh, generated from other systems. Here the results are especially impressive at corners since all known systems do not consider corners in their polygonisation routines.

Chapter 8

Results

In this Chapter, several examples for reconstructing triangular meshes from digitised data using the developed algorithm are given. All examples were processed on a PC under Windows NT 4.0 with a 750 MHz processor, 512 MB RAM and 267 MB extended memory. First, examples for the mesh generation from digitised data, provided from tactile, laser and camera based scanning systems are presented. The examples are supplied by different companies and have to be reconstructed for different applications. Some of the results in pre-processing applications such as milling or tool manufacturing are shown, as well. Also a comparison of the different examples in terms of computing time and number of triangles is given.

Then a comparison of the results from triangle decimation with different parameter values are given. It is shown, that even with small error bounds the number of triangles can be reduced significantly after the mesh generation with multiple view combination. It is also remarkable, that the geometry of an object is still recognisable even with larger tolerances.

Finally, some examples for the smoothness optimisation using edge swaps are given. The quality of the resulting surface, especially at corners or small radii can be improved significantly using the presented reorganisation scheme.

8.1 Examples for Mesh Generation

The presented examples (Figure 62-76) are parts, digitised for real industrial applications. All examples are optimised using the edge-swap optimisation. Also the number

of triangles has been reduced with $0.05mm$ standard deviation, 2^o angle tolerance and a maximum edgelength of $20mm$. In Table 1, a comparison in terms of the number of triangles ($TriNr$) before and after decimation ($TriNr - red.$), and computing time (without the time, required for triangle decimation) is given. The variety of objects shows the robustness and effectiveness of the presented approach. Even highly detailed models such as the *carrot man* (Figure 65), closed objects such as the *door handle* (Figure 73) or examples with an extremely high number of data points such as the *BMW1* with $\approx$ 19 million data points can be processed with the presented approach[1].

Example	Type	Scans	PoiNr	TriNr	Time	TriNr-red.
exhaust part	laser	24	597.037	600.458	9,1min	121.341
laser2	laser	15	275.105	316.428	3,9min	73.021
laser3	laser	7	487.119	487.430	5,2min	105.064
laser4	laser	67	8.916.662	1.308.558	39,9min	329.006
door handle	laser	12	833.302	943.215	30,0min	148.744
angel	laser	16	845.280	1.105.486	38,2min	280.612
beetle	photo	35	544.280	1.002.363	7,2min	332.543
carrot man	photo	7	837.845	1.004.522	76,4min	321.992
connecting rod	photo	8	202.785	367.084	2,6min	103.829
smart	photo	107	1.501.408	2.785.472	21,8min	761.606
bmw1	photo	50	19.090.356	2.858.138	591,6min	845.369
bmw2	photo	35	6.022.532	4.063.812	152,8min	565.433
bone	photo	25	487.119	424.761	12,5min	219.911
tooth	tactile	2	162.077	141.874	0,5min	33.440
sheet metal	tactile	4	83.269	157.618	0,4min	25.124

Table 1: Statistics for the reconstruction of different models.

The triangle decimation approach allows a reduction of about 1000 triangles per

[1]The bmw1 example consist of $\approx$ 19 Million data points, which could not be processed in total. Therefore a triangle decimation before limiting has been applied to each single mesh. The parameter were chosen, such that only a small error was allowed ($DeviationTol = 0.03$, $AngleTol = 2^o$) and that no large triangles are generated ($Maximumedgelength = 2 * LineStep$). For each view, the number of triangles could be reduced by a factor of ≈ 3 which allowed the object to be reconstructed.

second. This means 100000 triangles can be eliminated in 1.5 minutes. All examples can be further prepared and processed in applications such as milling or surface reconstruction.

The time involved in the combination process depends on the number of overlapping views and overlapping triangles. In the example *carrot man*, almost all views do overlap each other. Therefore any view has to be compared to all others. Many triangles are removed and the resulting mesh consists of almost the same number of triangles as the *beetle* example. Here, the views do overlap only slightly and therefore the computing time is much smaller than in the carrot man example.

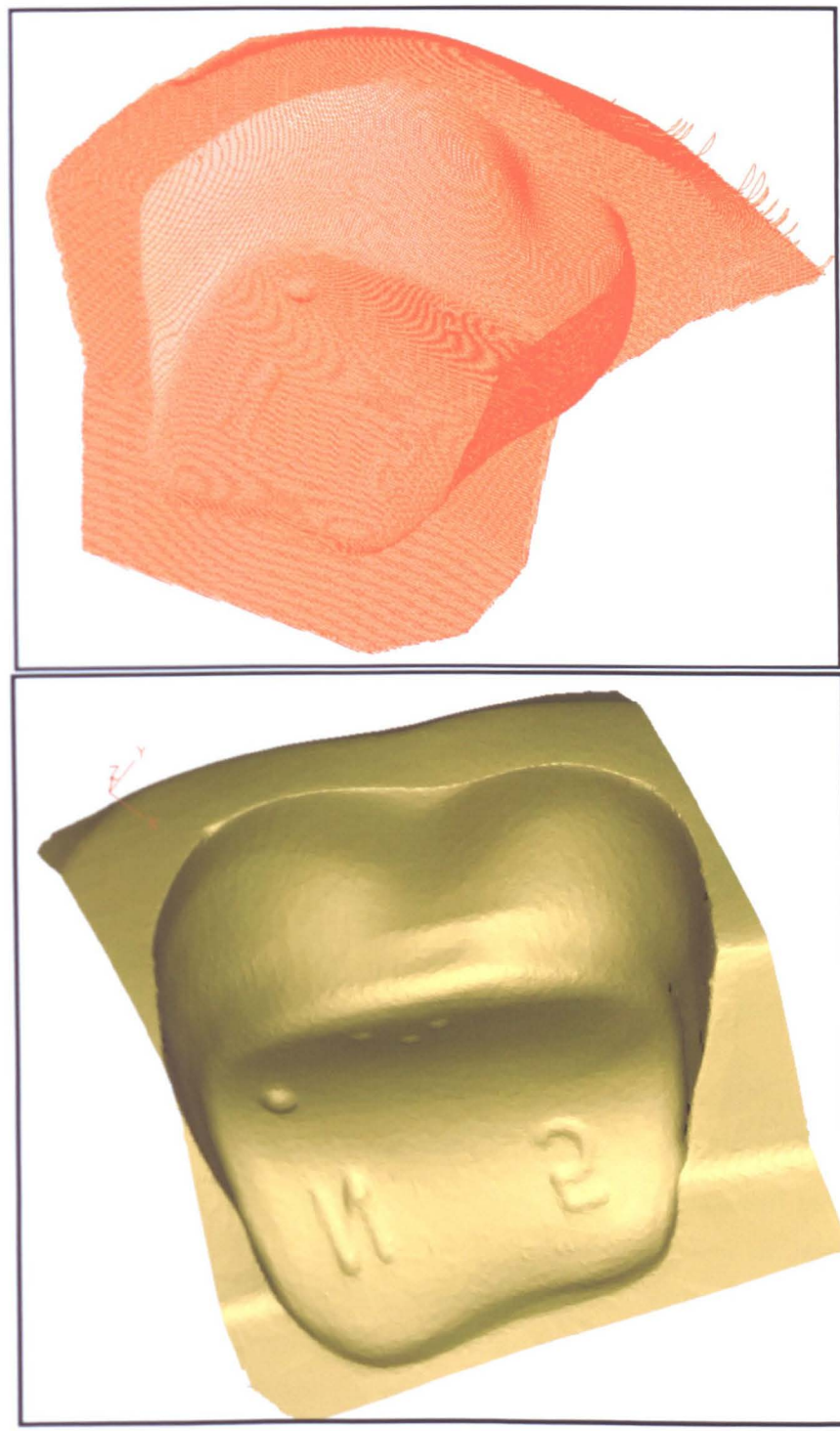


Figure 62: Example of a tactile measured *tooth*.

The *tooth* example (Figure 62) consists of two overlapping digits. It is scaled by a factor of 10 before polygonisation in order to apply the same error bounds as in the other examples.

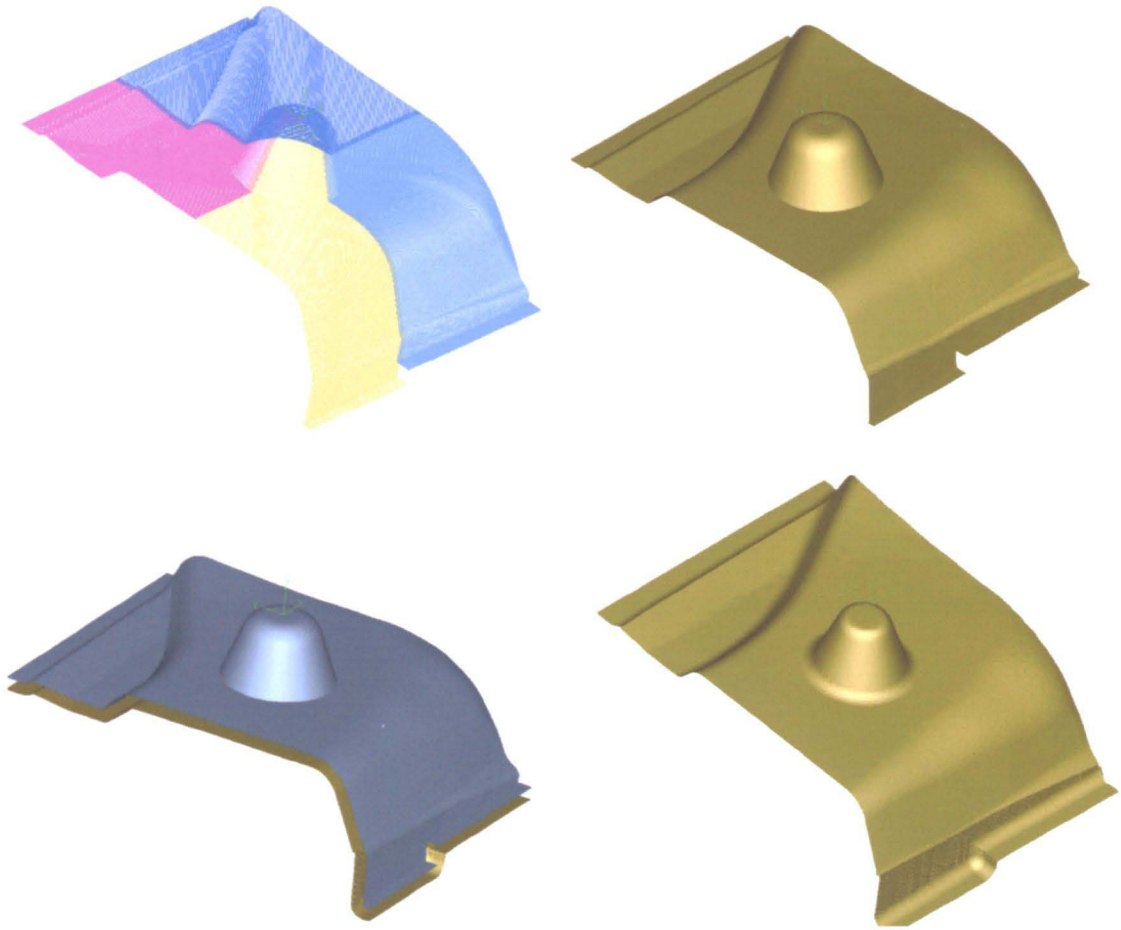


Figure 63: Example of a tactile measured *sheet metal.*

The *sheet metal* example (Figure 63) is measured from four different point of views using a tactile measuring machine. The top left figure shows the digitised data. The top right is the reconstructed surface. Since the object is digitised using a scanning probe with radius $4mm$, this radius has to be compensated. The middle left and right pictures show the result after this radius compensation. Note, that the sharp corner around the dome is reconstructed as a blending after offset compensation (middle left figure) This blending might have been a sharp corner in the original part, so a pre-processing step is necessary to reconstruct this feature. Some examples including the *beetle* example are provided by the company GOM, Gesellschaft für optische Messtechnik, D-38106 Braunschweig, Germany.

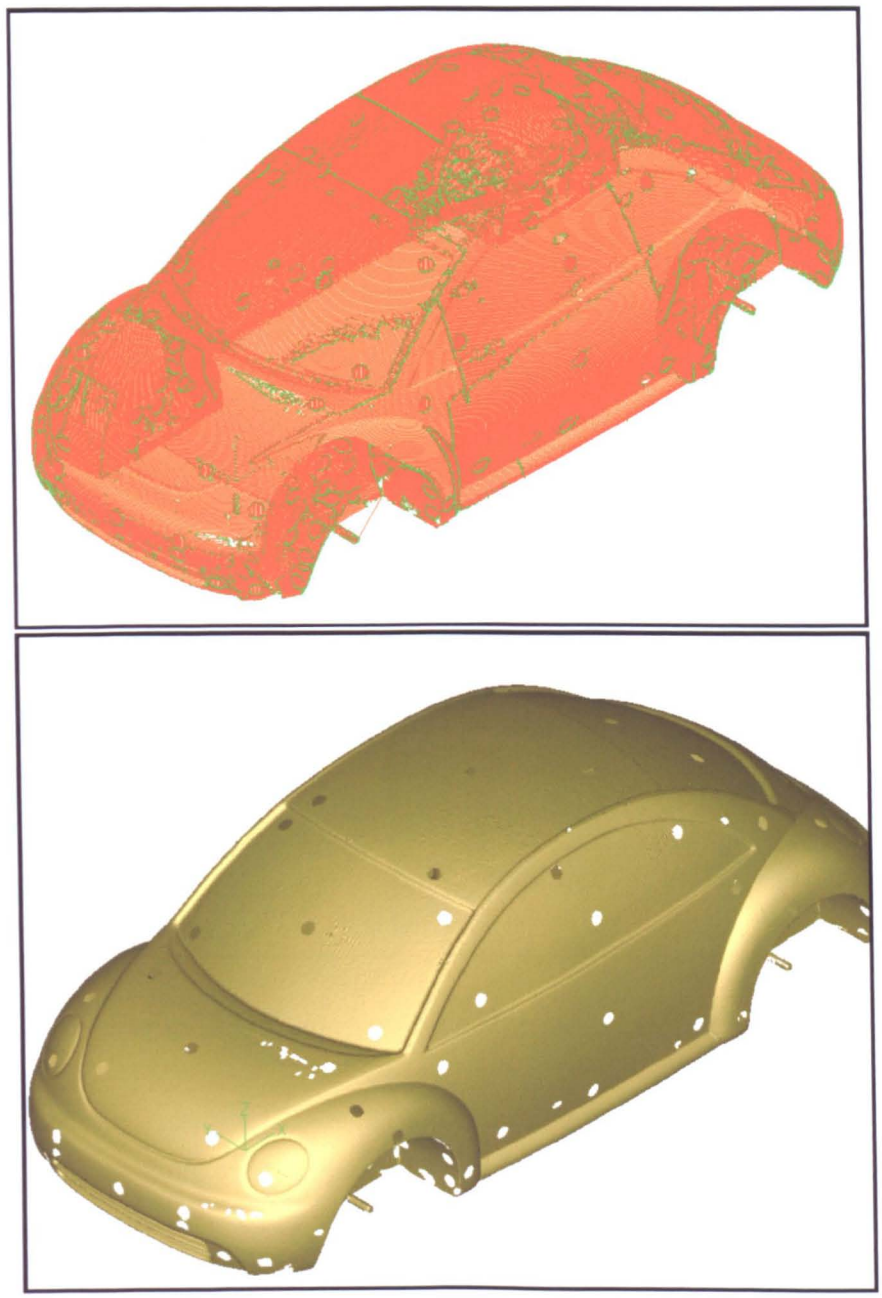


Figure 64: A reconstructed beetle, digitised with structured lighting.

In the *beetle* example (Figure 64) there are holes, which remain from the basing marks. They can easily be filled using a simple hole-filling algorithm (based on simple polygon triangulation), since they are in almost planar regions.

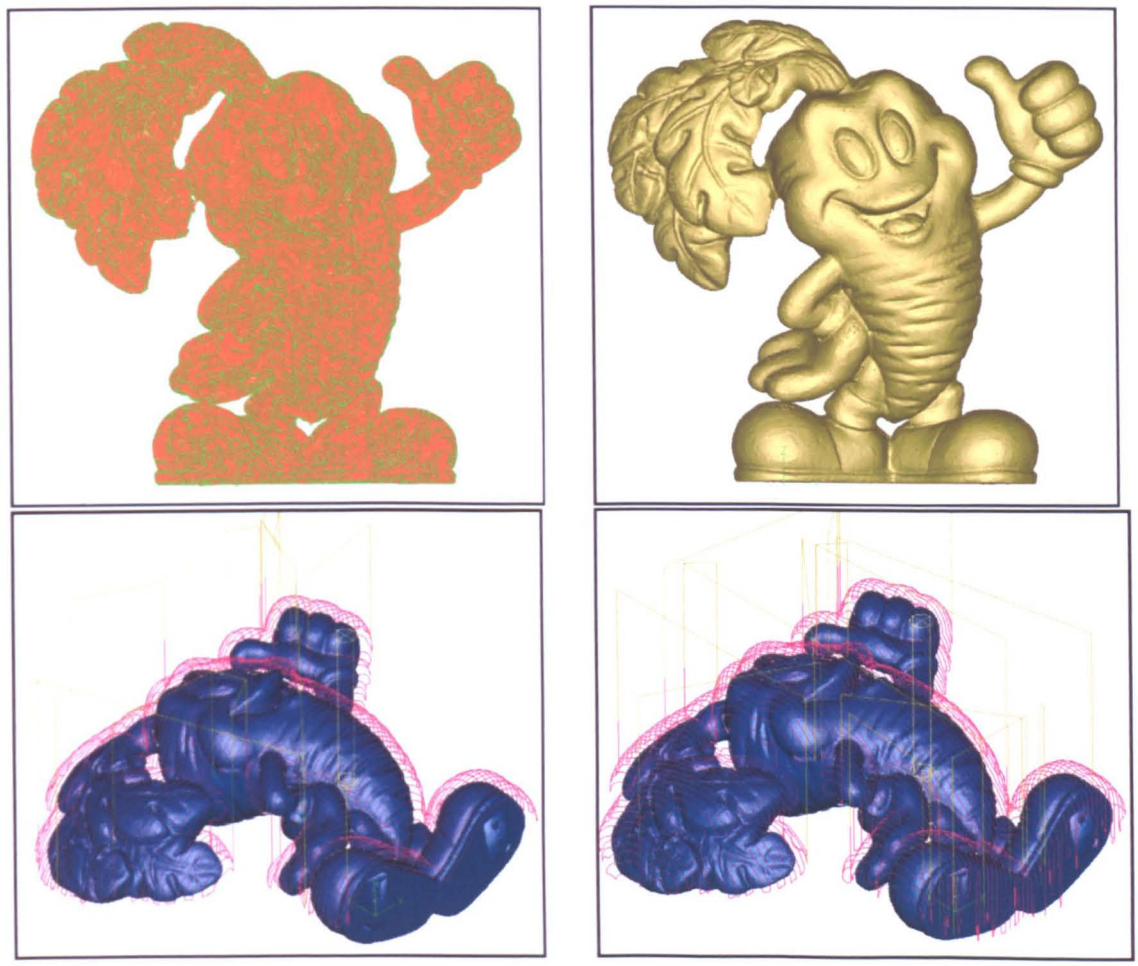


Figure 65: Example of a polygonised *carrot man*.

The *carrot man* example (Figure 65) is obtained from the Internet. It differs from the other examples, because it consists of many completely overlapping views, which leads to many removable triangles and to long computing time. The bottom two views show the toolpathes, generated to reconstruct the object on a milling machine.

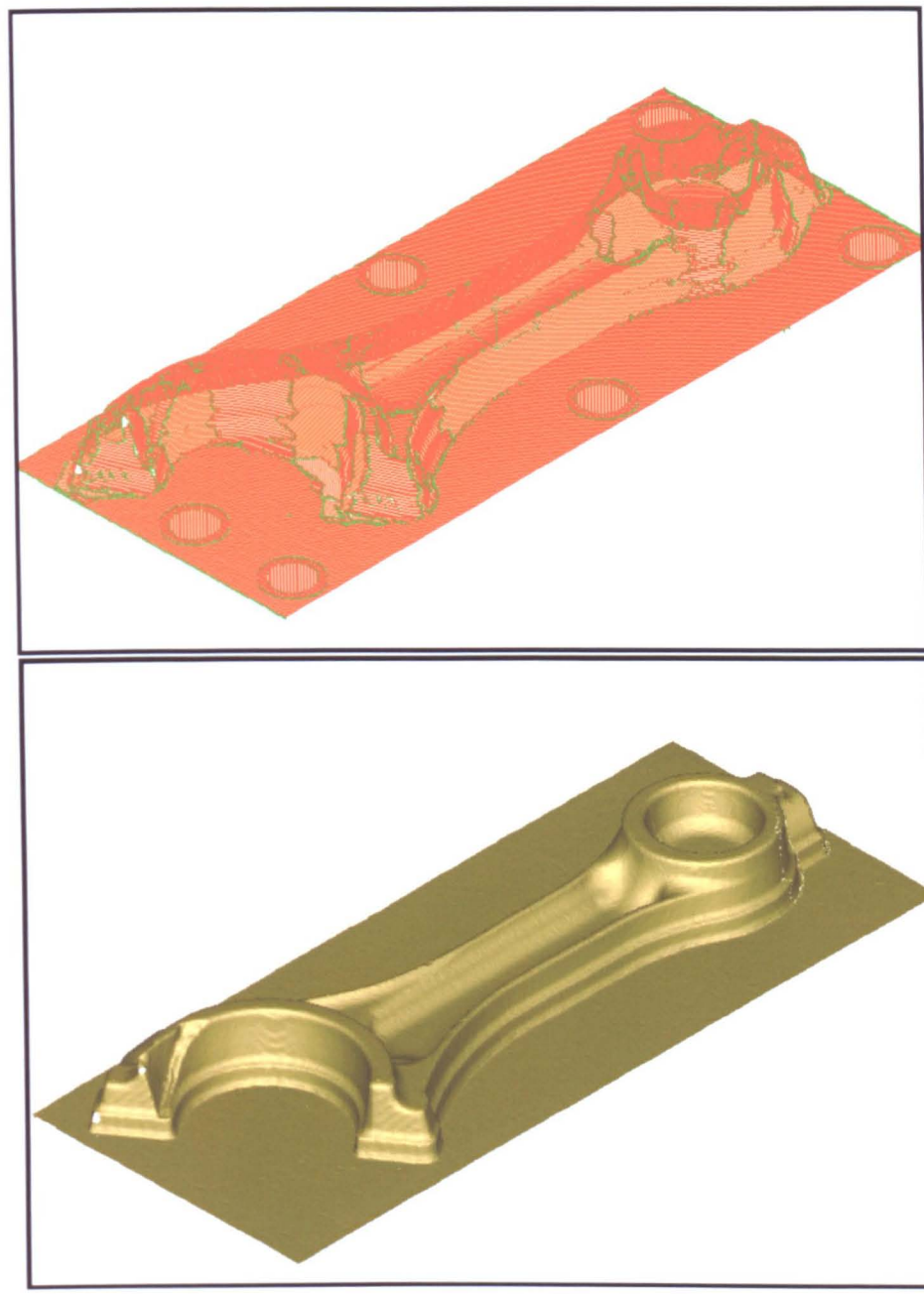


Figure 66: Polygonisation of a *connecting rod.*

In the *connecting rod* example, the holes due to the basing marks were filled in advance within the digitising system. Unfortunately, there is no view direction given for these basing marks. The single meshes, generated from these hole fillings are limited but not connected automatically. In this case it would be desirable to assume directions of view for the fill-areas in order to process them equally to the other views.

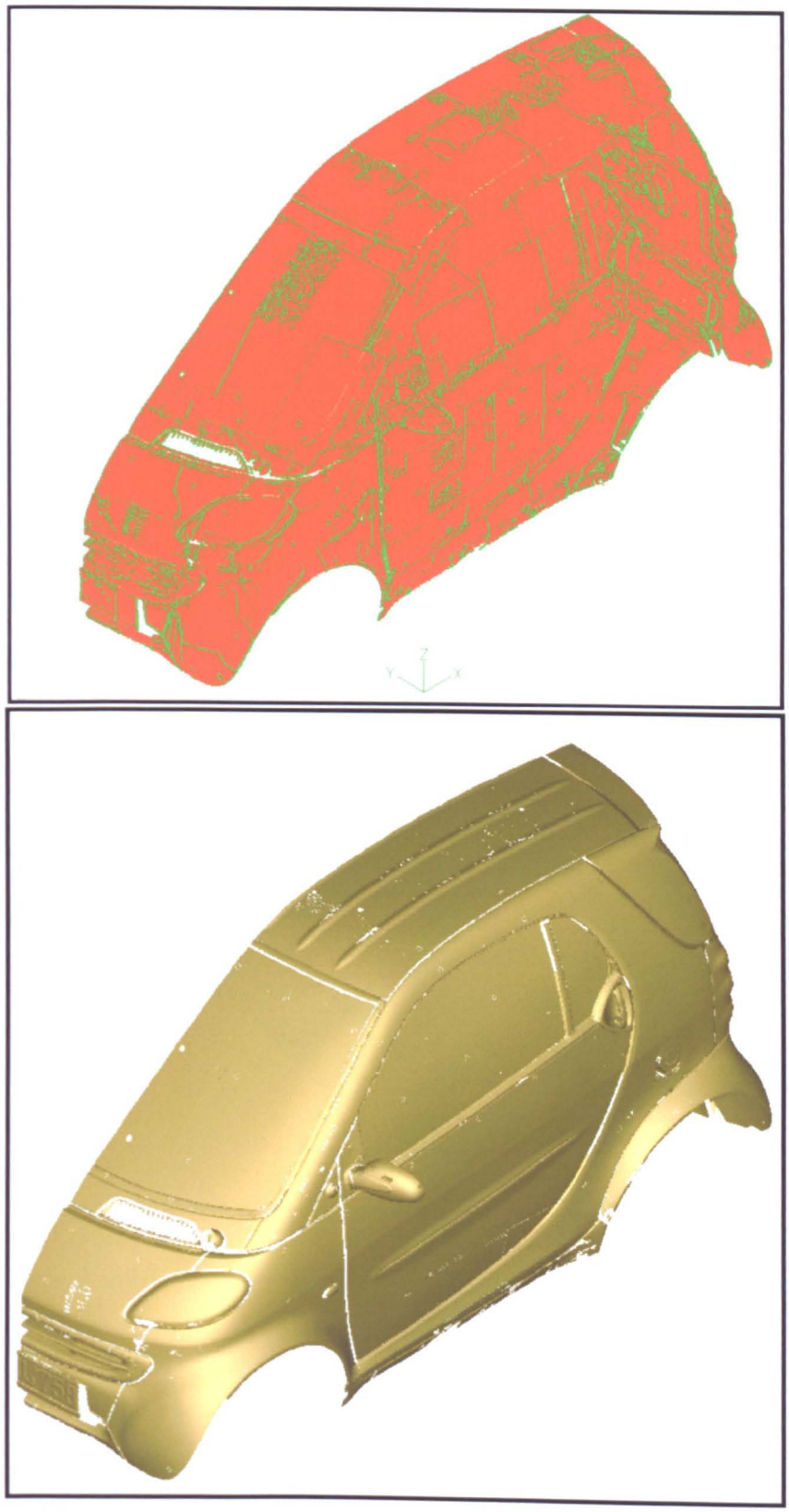


Figure 67: Example of a digitised *Smart*.

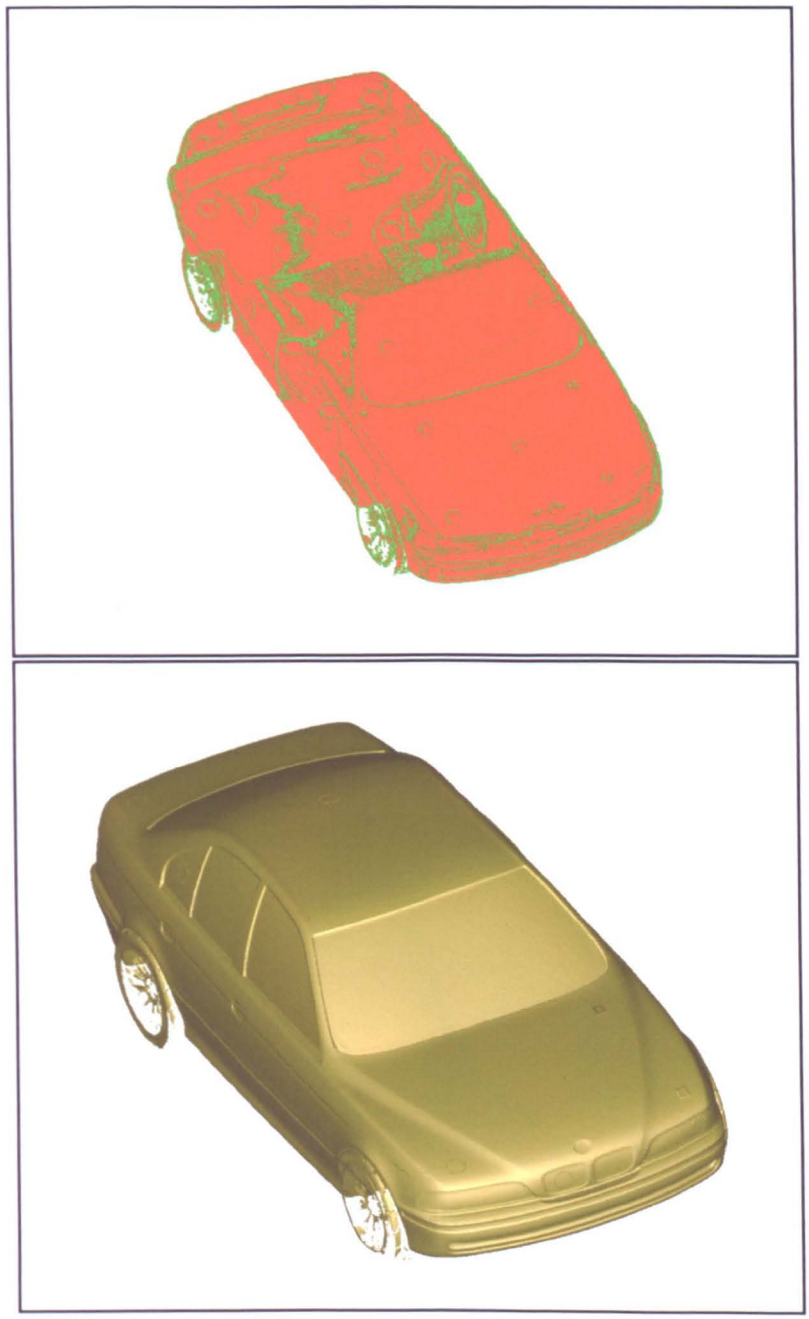


Figure 68: Example of a reconstructed *BMW (1)*.

The *BMW (1)* (Figure 68) is the example with the highest number of data points. Considering 12 Bytes for three floating point numbers X, Y, Z lead to $240MB$ required storage, only for the data. This example could not be handled without pre-filtering. We therefore applied triangle decimation prior to the combination to each view.

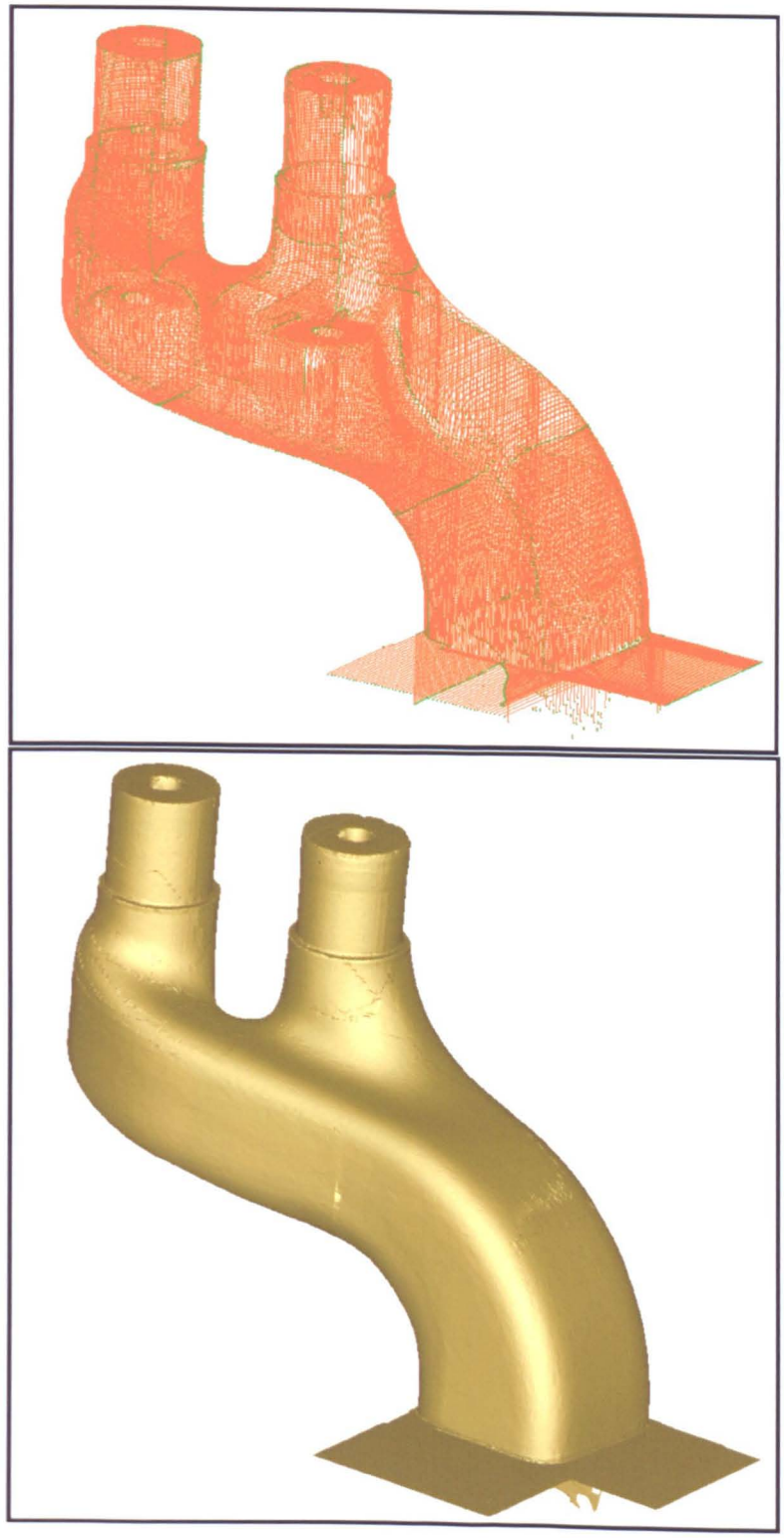


Figure 69: Example of a laser scanned *exhaust pipe*.

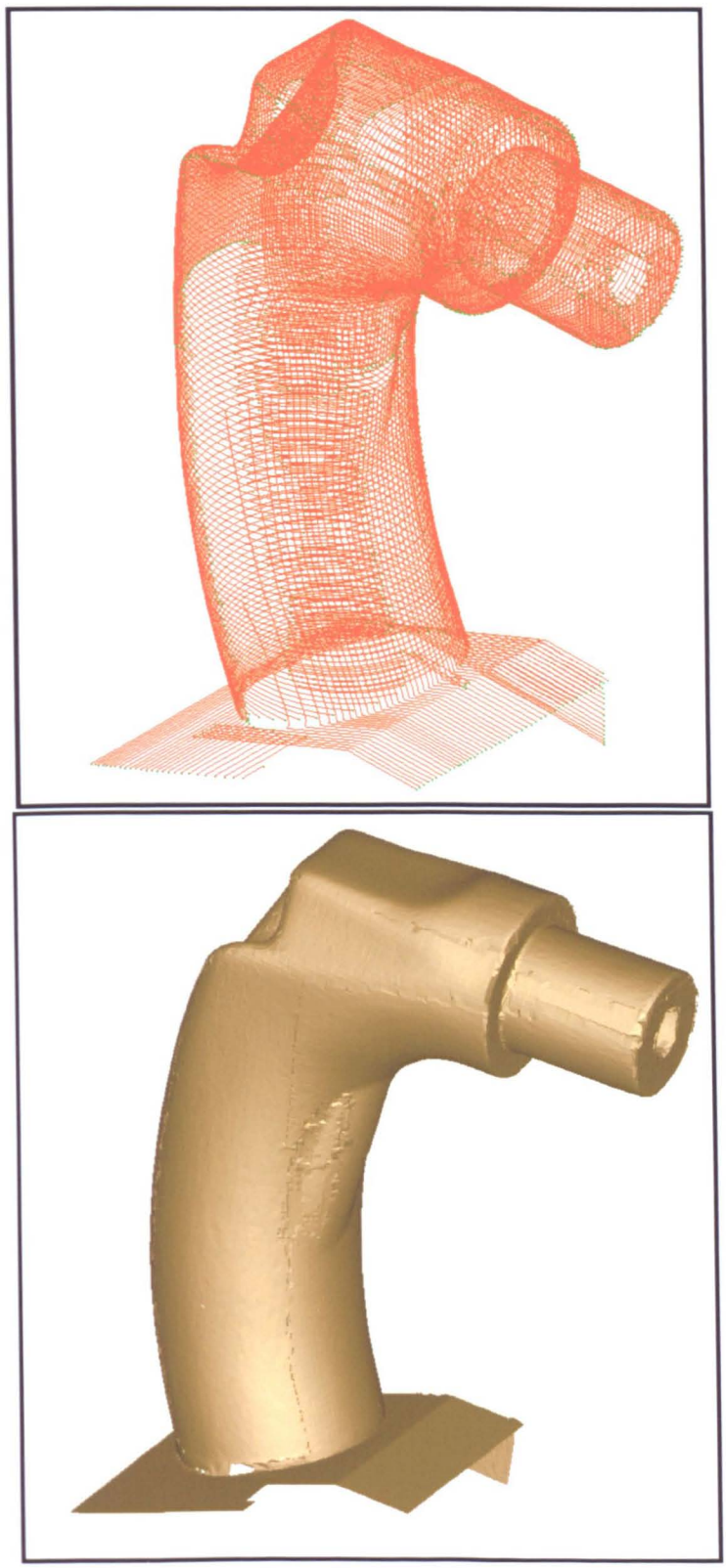


Figure 70: Example of a laser scanned mechanical part.

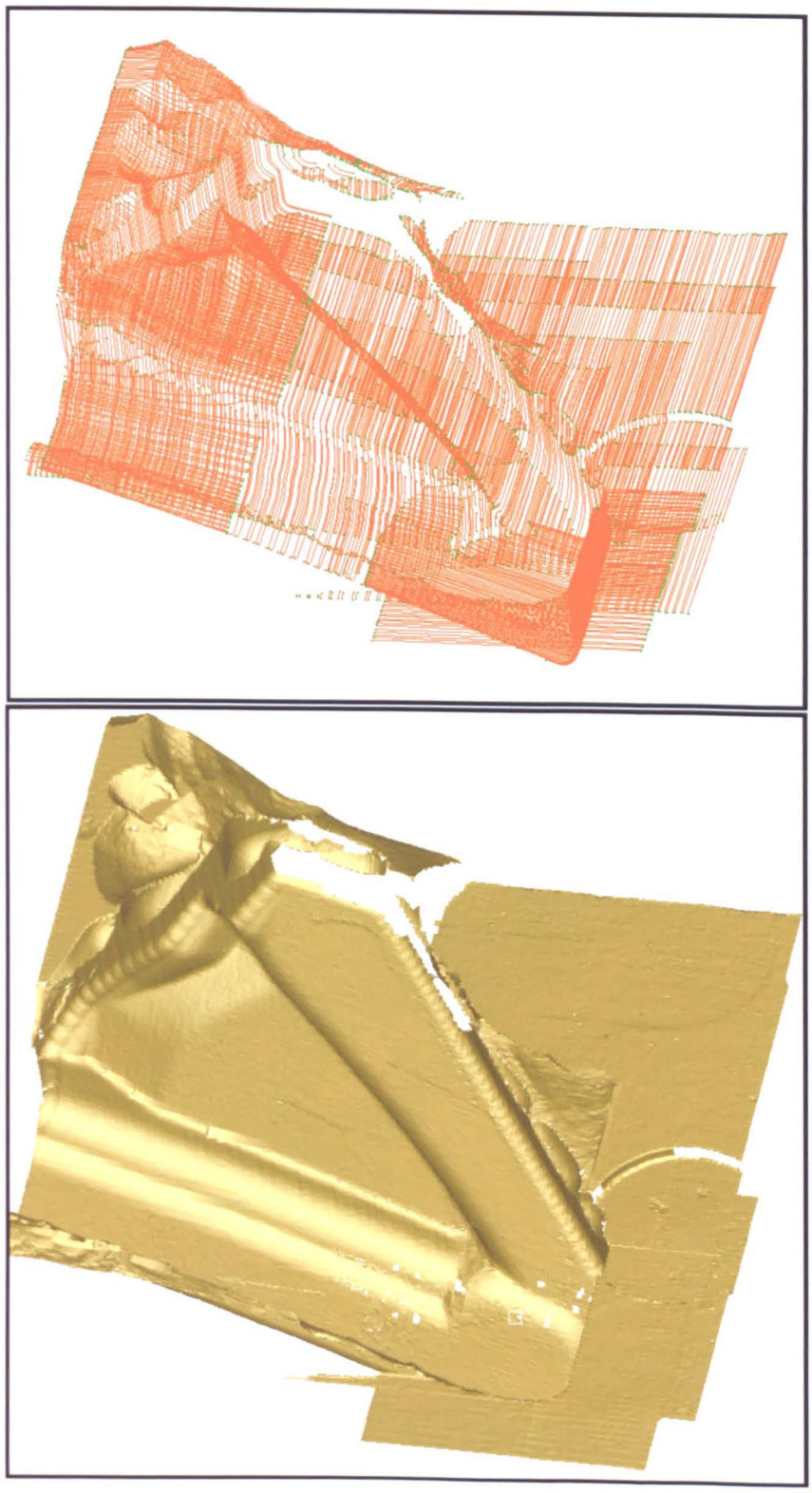


Figure 71: Example of a laser scanned compression die.

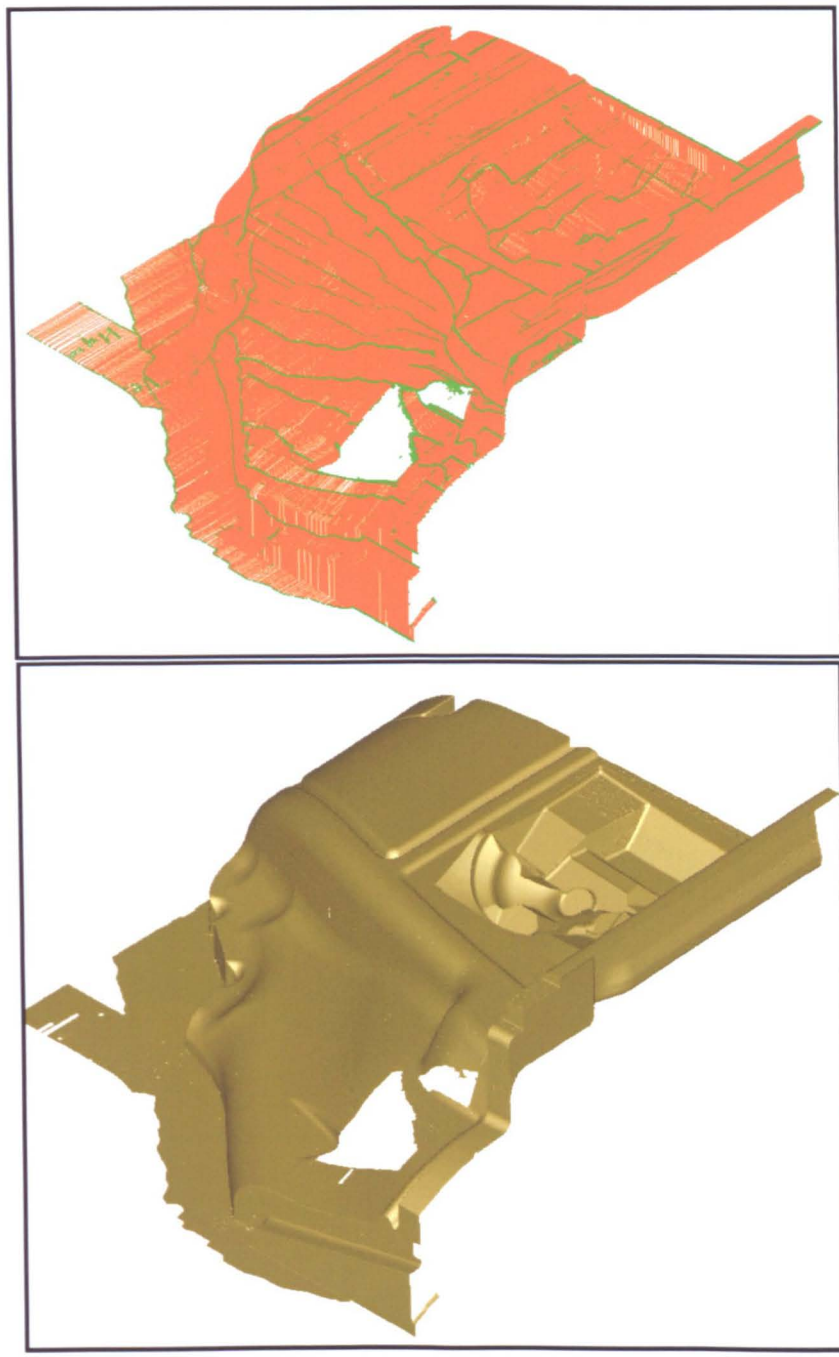


Figure 72: Example of a laser scanned interieur part of a car.

This example consists of about $9Mio$ data points. The laser scans the object with a point resolution of $0.1mm$ in the row. The number of data points can be reduced in advance using a scanline filtering approach with $0.03mm$ deviation tolerance, 2^o angle tolerance and a minimum point resolution of $1mm$. In this case, this filtering leads to a much higher performance while the object is still reconstructed nicely.

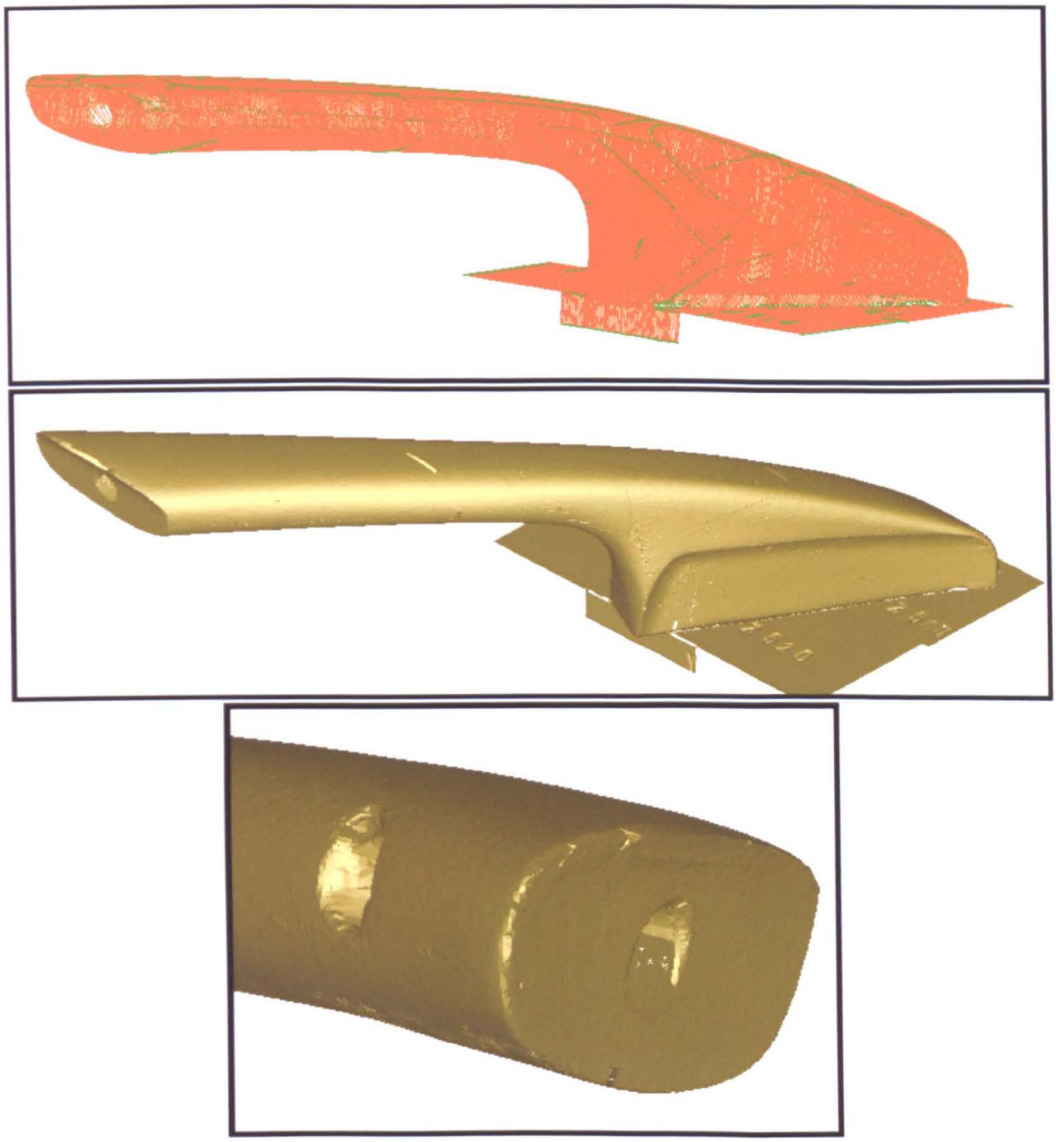


Figure 73: Example of a laser scanned *door handle*.

Note, that there is a hole in the part, where the data object could not be digitised and which therefore could not be reconstructed due to missing data. Since this hole is of regular geometry, it can be reconstructed within a CAD system using spline surfaces and a cylinder function. Still, the problem of integrating this cylinder into the mesh has to be solved. Intuitively, this operation might be a boolean solid operation, since the presented meshes are 2-manifolds.

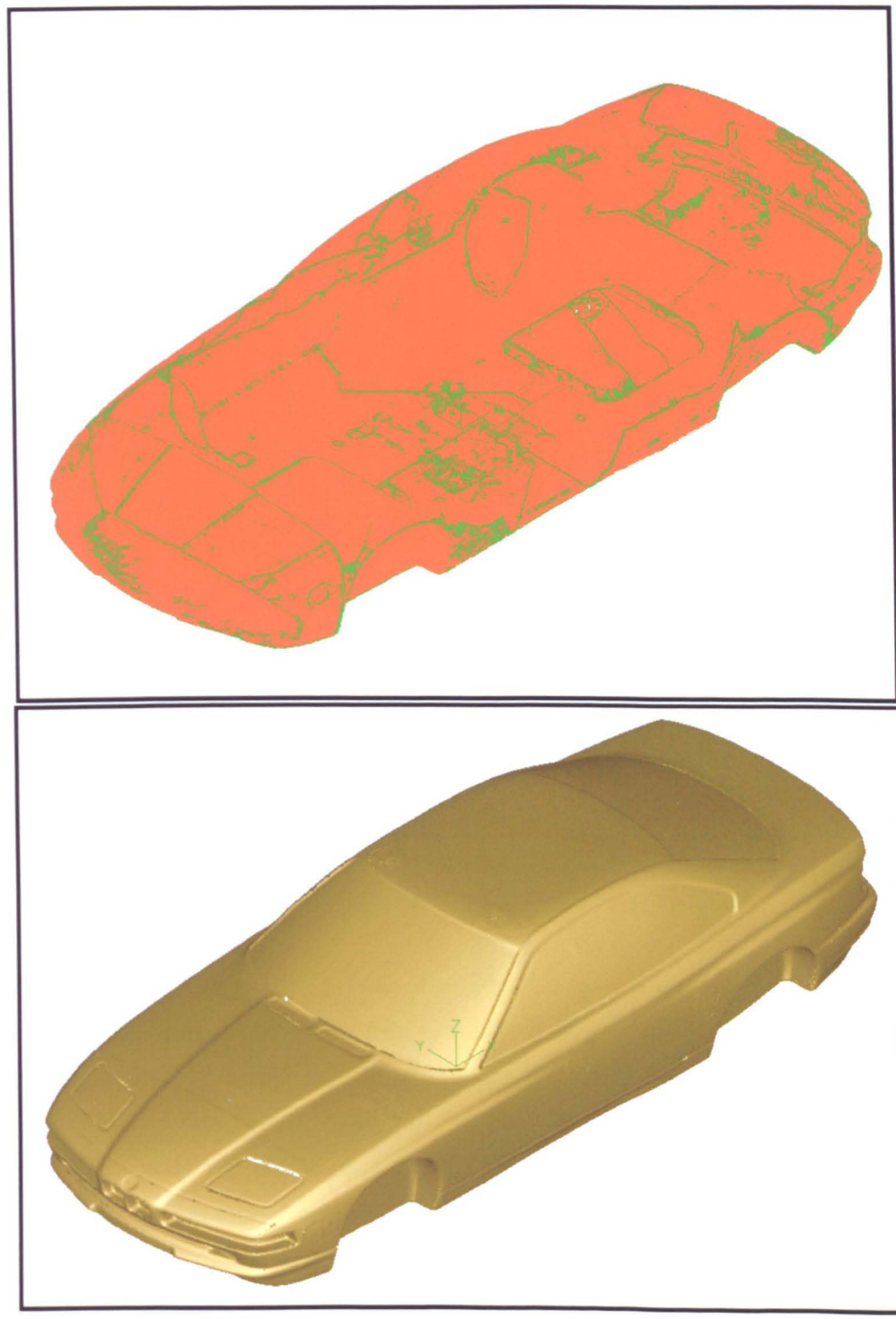


Figure 74: Another *BMW (2)*, digitised with photogrammetry.

The BMW(2) example was digitised using the *Comet* scanner from Steinbichler Optotechnik, D-83115 Neubeuern, Germany.

Figure 75: Example of a digitised *angel statue.*

The *angel statue* (Figure 75) was digitised using a *body scanner.* (Vitronic Dr.-Ing. Stein, Bildverarbeitungssysteme GmbH, D-65189 Wiesbaden, Germany.) The scanner consists of 16 range lasers, which are mounted on a common travelling device. The device then moves from the bottom to the top of the object, capturing the hole object with all lasers at a time. The capturing only takes $\approx 15sec.$ This is fast enough to capture human *bodies.* There is an interesting project running in the Netherlands called the CAESAR project for determining the body dimensions of Dutch adults, using the bodyscanner. Information can be obtained from *www.nedscan.nl/nedscan/startuk.htm.*

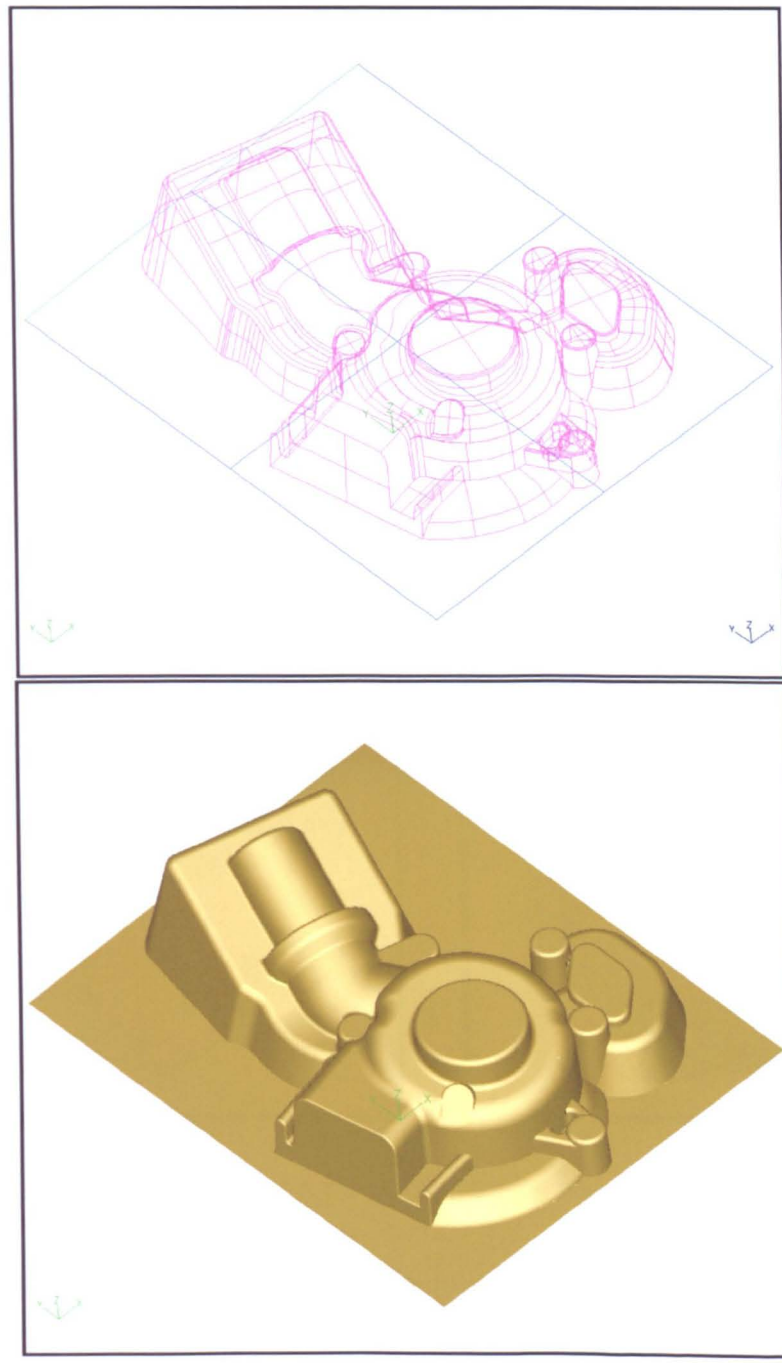


Figure 76: Example of a *pump*.

The *pump* example (Figure 76) has not been digitised, but generated using an ax-parallel CAM milling application. Two NC-toolpathes are generated, one parallel to the x-axis and one parallel to the y-axis. Both programs are fully overlapping. The toolpathes are then converted into digits and processed equally to digitised data. Note, that all sharp corners and small radii are well presented. Since there is no noise in the data, the surface of the part is nice and smooth.

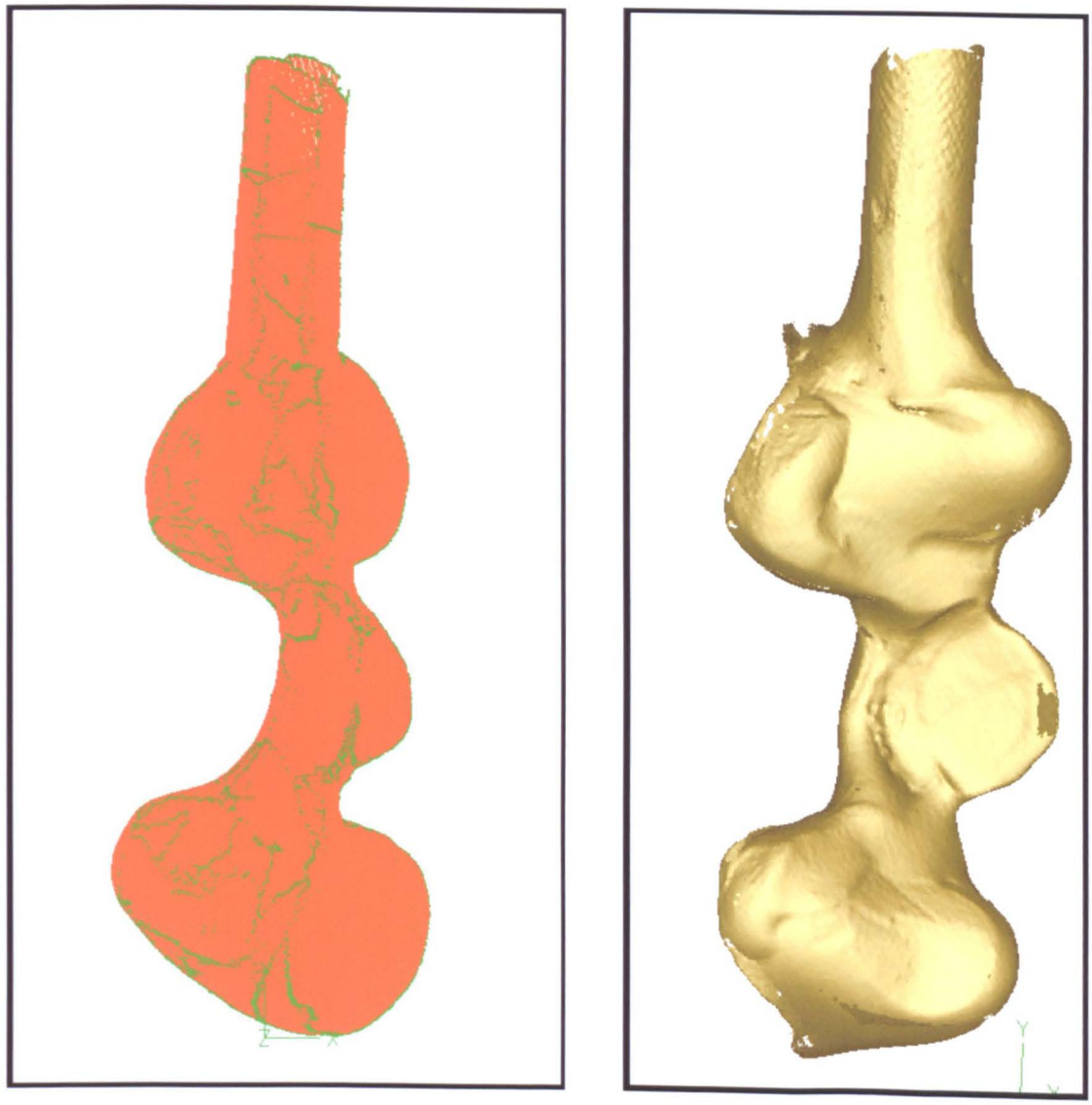


Figure 77: Example of a *bone*, digitised with photogrammetry.

A part such as the *bone* (Figure 77) can hardly be reconstructed with polynomial surfaces since the geometry is too complex. Reconstructing this object with triangular meshes is easy, quick and the geometry is well presented.

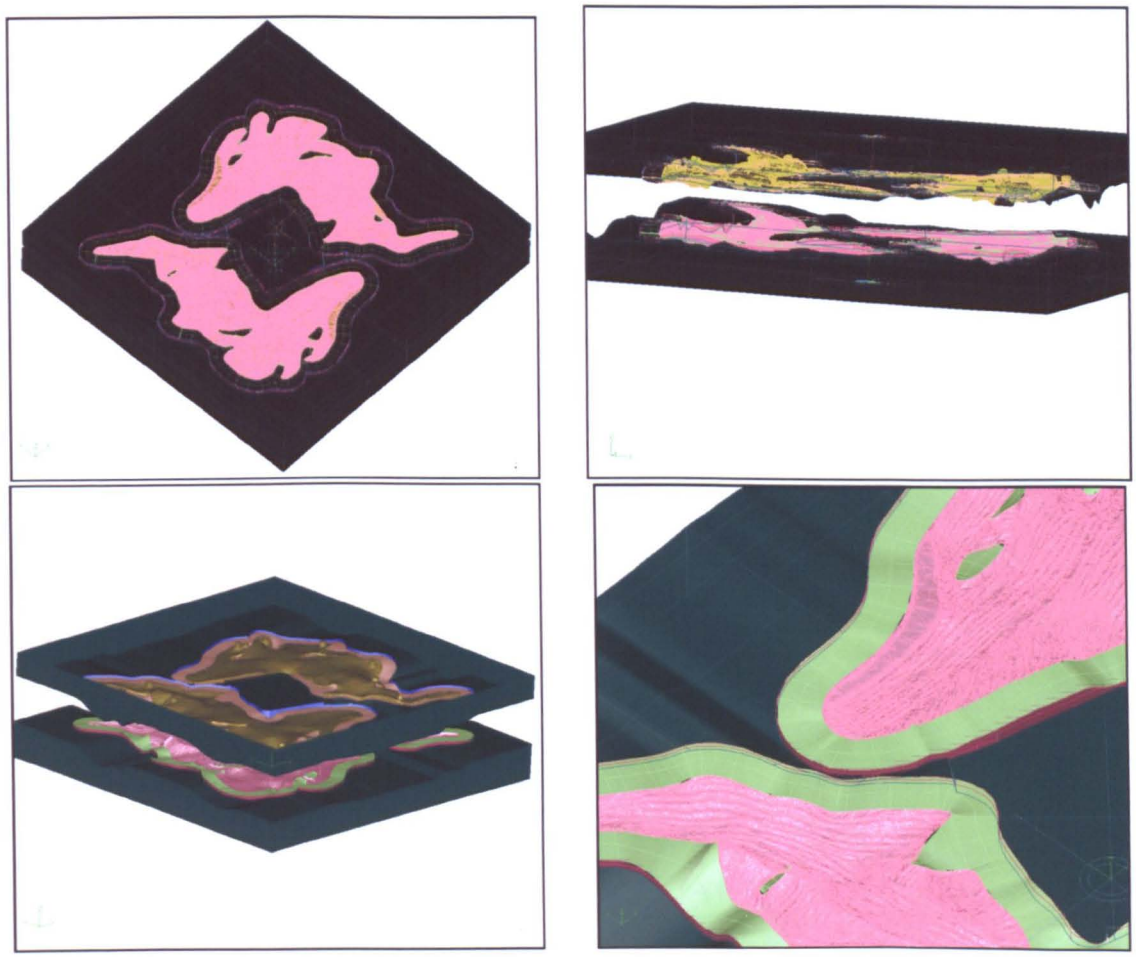


Figure 78: Hybrid model, consisting of polynomial surfaces and triangular meshes.

The final example is a complete compression moulding die (Figure 78), consisting of triangular meshes and CAD surfaces. The company 3bScientific provided this example. They are specialised in the manufacturing of anatomical and biological teaching aids for science, training and patient education. Information can be obtained from *www.3bscientific.com*.

8.2 Examples for Triangle Decimation

Triangle decimation is used in order to reduce the number of triangles and therefore to minimise storage and speed up computing time in pre-processing applications.

Here some examples (Figure 79-82) are presented and different parameter values for deviation tolerance (*DisTol*) and angle tolerance (*AngTol*) are compared. The *DisTol* depends on the specific accuracy of the scanner. It should be chosen such that it is slightly larger than the scanner accuracy. Larger values are only recommended, if the number of triangles must be reduced to a specific minimum number, or if the expected reconstruction tolerance is higher. The angle tolerance does not depend on the part dimensions or the scanner resolution and can be chosen by the user.

Connecting rod	185.473	0.03mm	2°/ 50.155	15°/21.585	45°/ 14.557
	185.473	0.1 mm	2°/ 38.287	15°/11.722	45°/ 3.803
	185.473	0.5 mm	2°/ 2.933	15°/ 2.479	45°/ 1.204
Sheet metal	22.410	0.03mm	2°/ 12.117	15°/ 2.305	45°/ 2.112
	22.410	0.1 mm	2°/ 5.193	15°/ 984	45°/ 773
	22.410	1.0 mm	2°/ 2.644	15°/ 424	45°/ 291
Beetle	1.002.363	0.03mm	2°/550.647	15°/314.912	45°/291.977
	1.002.363	0.1 mm	2°/164.921	15°/ 77.752	45°/ 57.823
	1.002.363	1.0 mm	2°/122.267	15°/ 36.220	45°/ 13.314

Table 2: Statistics for triangle decimation.

The results for the connecting rod example are shown in Figures 79, 80 and 81. Note that even with higher tolerances, the resulting mesh is still quite near to the original. This is due to the specific ordering, in which vertices are removed from the mesh. The vertices removed first are almost the same with all different parameters, since the quality for removement and therefore the priority does not depend on the specified tolerances. The recommended parametervalues in this case are $0.1mm$ deviation tolerance and 2° angle tolerance. The number of triangles can then be reduced from 185.473 to 38.287 triangles, which is a decimation by a factor of $\approx 1/5$.

Table 2 gives an overview of two more examples. Figure 82 shows the *beetle*, reduced from 1.002.363 to a number of 13.314 triangles.

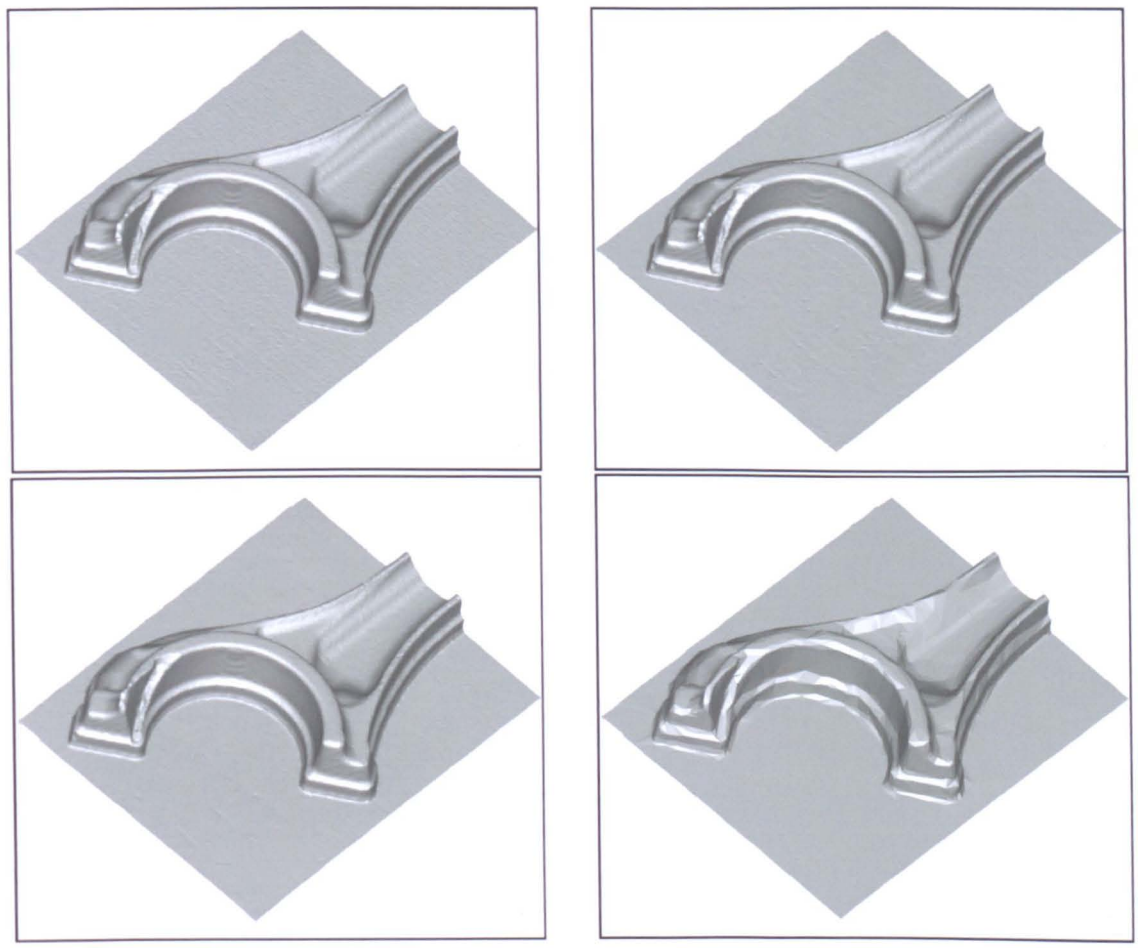


Figure 79: Example for triangle decimation, Part I of III.

This example is a part of the *connecting rod* example. The top left picture shows the part with 185.473 triangles. On the top right, the example was filtered with $AngleTol = 2°, DeviationTol = 0.03mm$, which leads to 50.155 triangles. Bottom left was filtered with $AngleTol = 2°, DeviationTol = 0.1mm$, leading to 38.287 triangles. The example on the bottom right was filtered with $AngleTol = 2°, DeviationTol = 0.5mm$. This example consists of only 2.933 triangles.

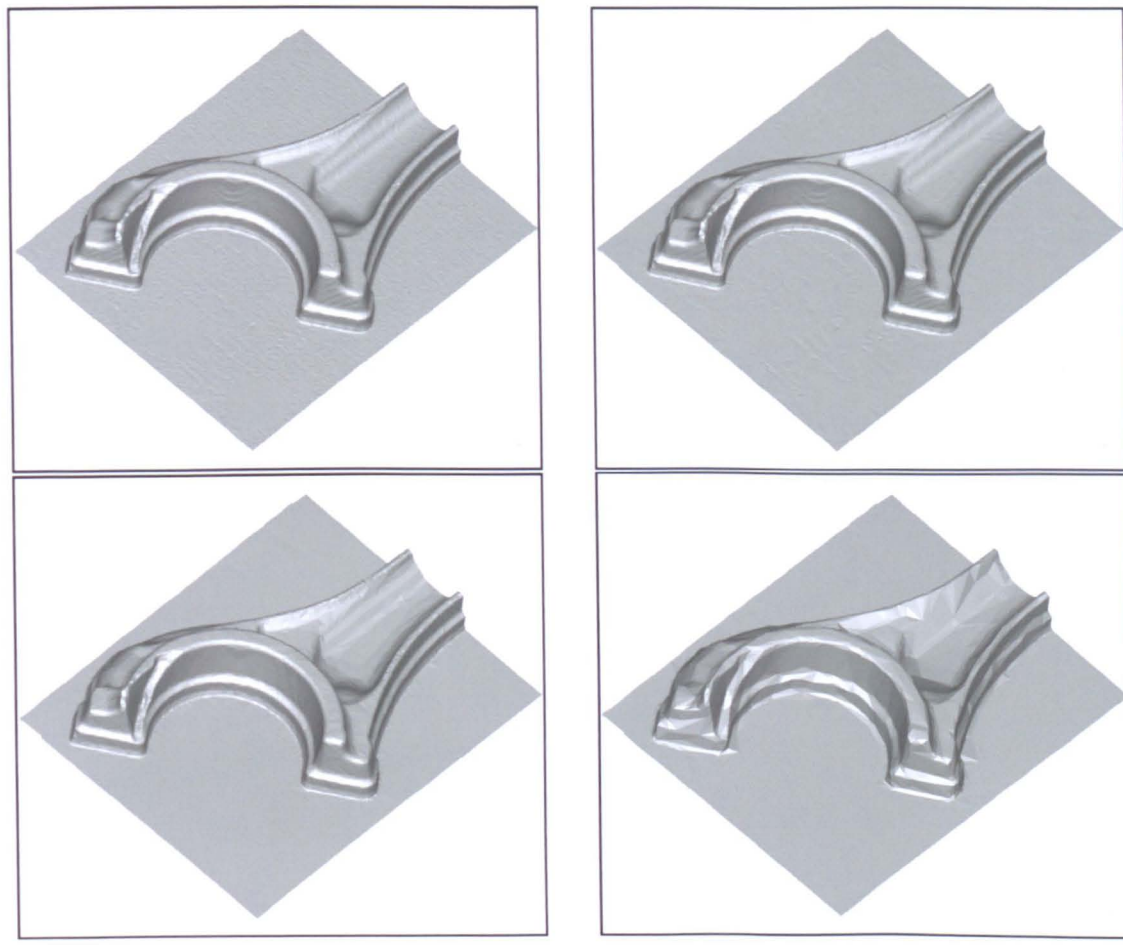


Figure 80: Example for triangle decimation, Part II of III.

The example was filtered with the same deviation tolerances, but with an $AngleTol = 15°$.

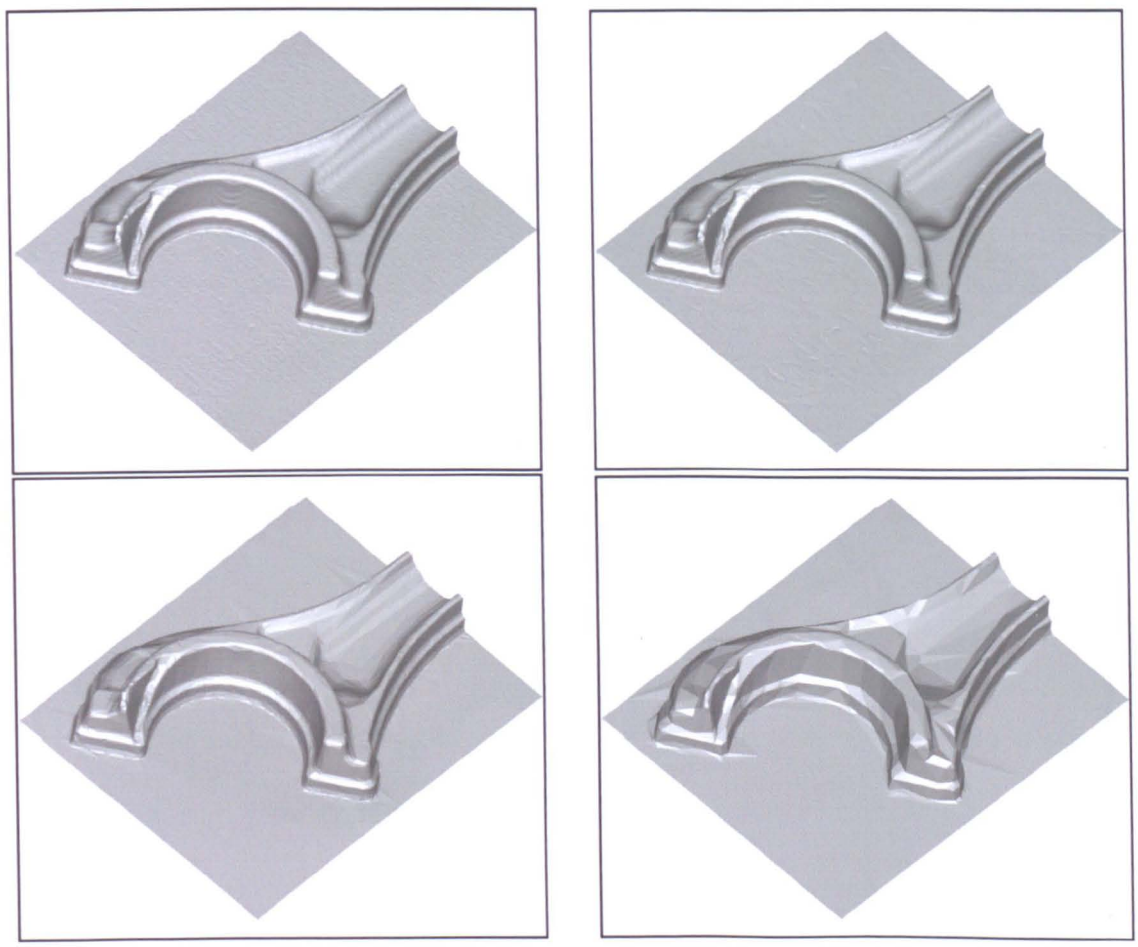


Figure 81: Example for triangle decimation, Part III of III.

In this final examples, the connecting rod was filtered with an $AngleTol = 45°$. Note, that even with the maximum tolerances of $0.5mm$ and 45° on the bottom right, the object is still recognisable and sufficient for visualisation. The number of triangles is only 1204. Of course, a much higher graphic performance can be achieved with this low number of triangles, even if the object does not meet tight tolerances towards the given data points.

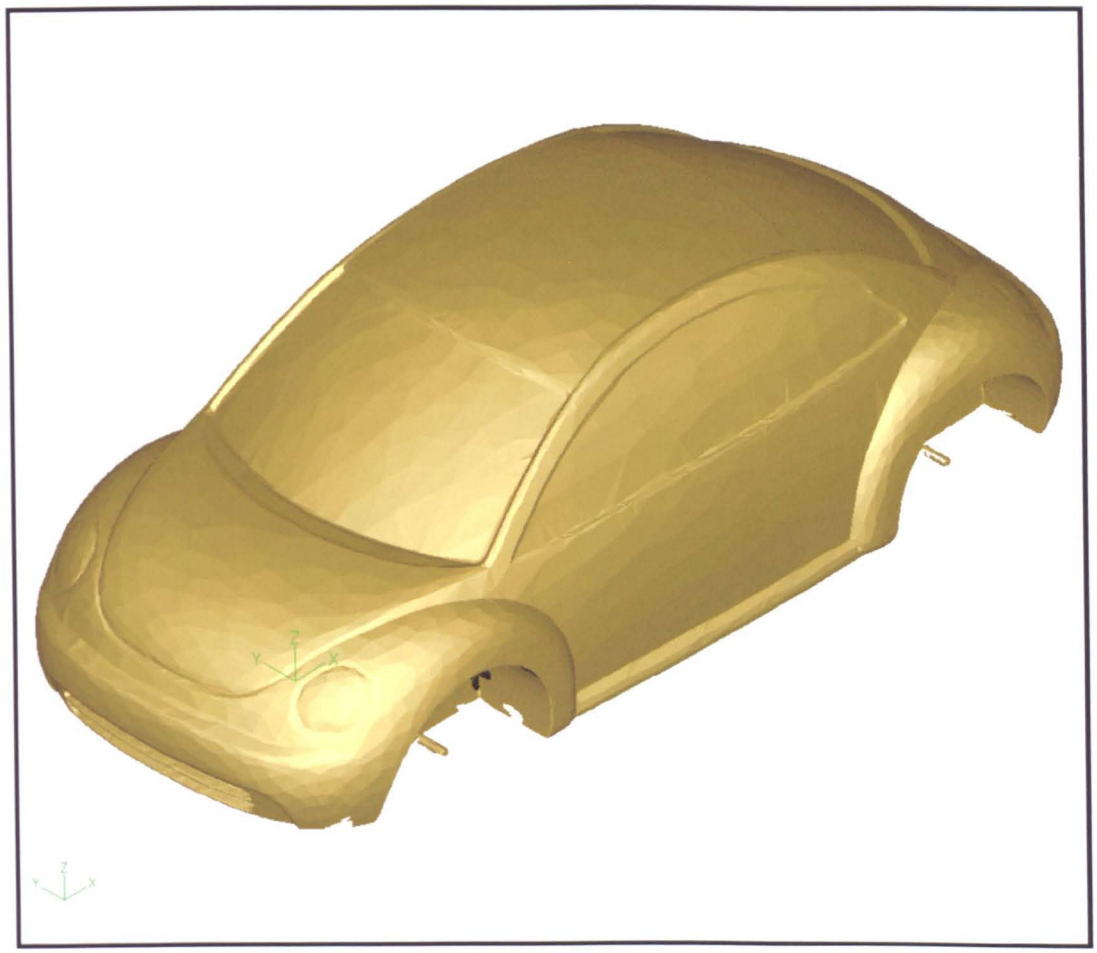


Figure 82: Triangulation of the *beetle* example, reduced to 13.314 triangles.

8.3 Examples for Mesh Optimisation using Edge Swaps

The quality of the reconstructed surface can be significantly improved using the simple edge swap optimisation, presented in section 4.3. On the following pages, some examples are given in order to illustrate the effect of the optimisation.

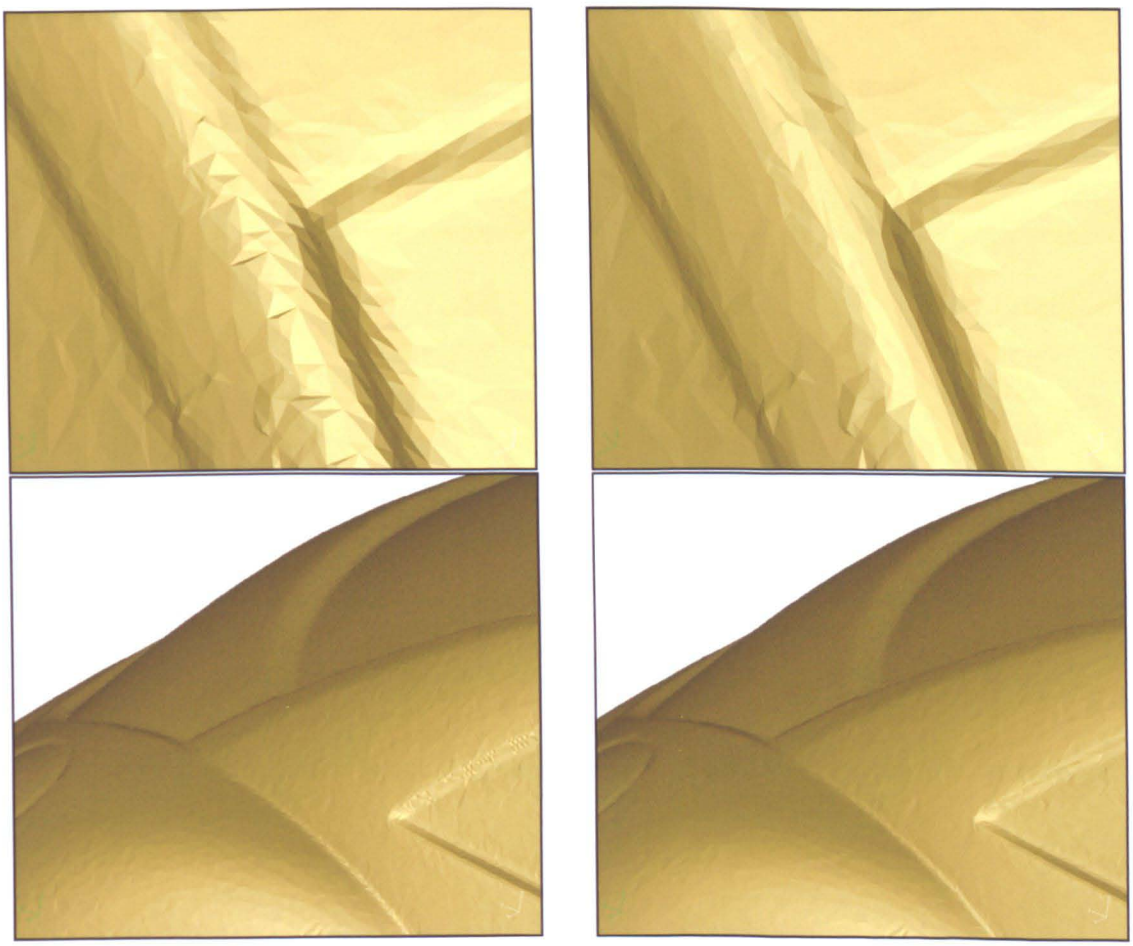


Figure 83: Examples for surface smoothing using edge swaps Part I of II.

Figure 83 shows parts of the *beetle* example, which is shown in Figure 64. On the left side, the example is shown before edge swap optimisation. The result after optimisation is shown on the right.

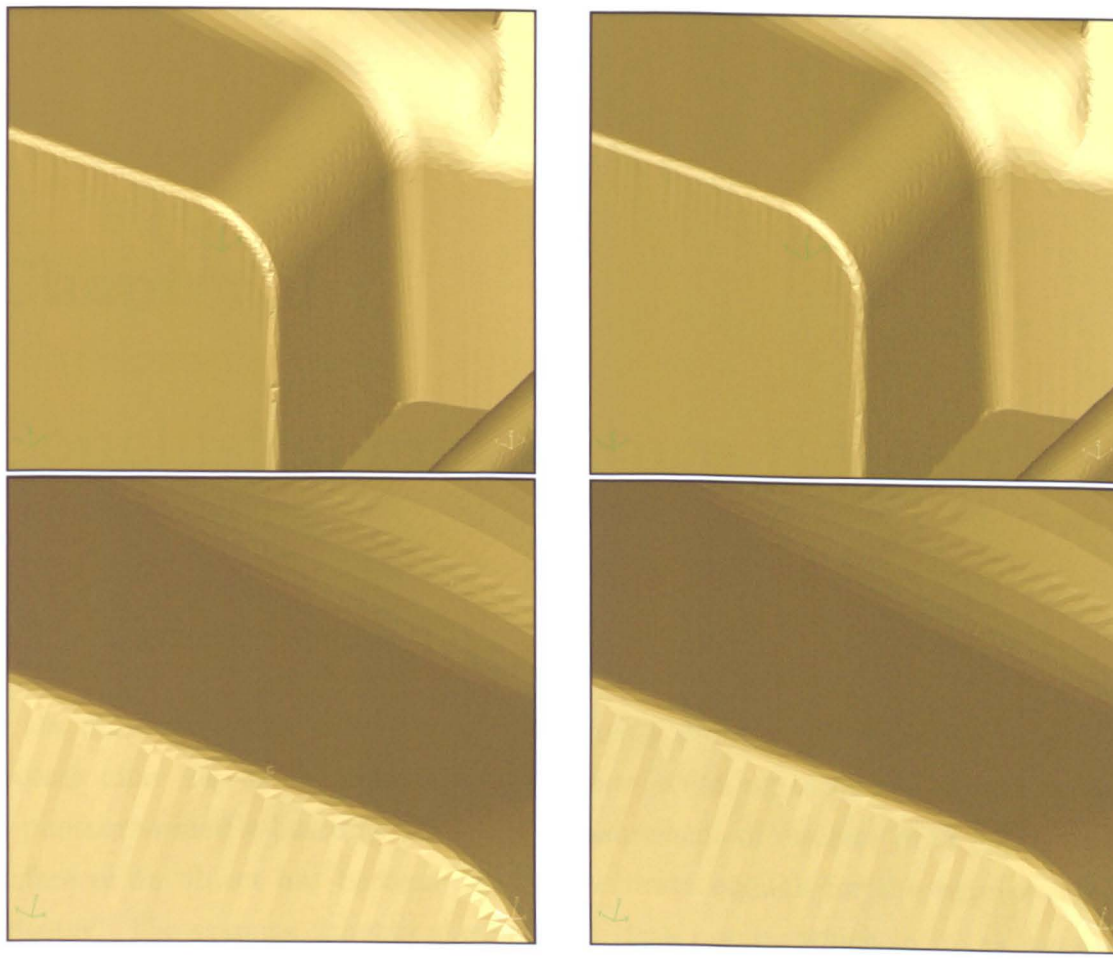


Figure 84: Examples for surface smoothing using edge swaps Part II of II.

The second example (Figure 84) shows parts of the *pump* example, presented in Figure 76.

Chapter 9

Conclusion and Future Research

9.1 Conclusion

In the past years, the conversion of real physical objects into computer models has become more and more important. With new scanning technologies such as laser or photogrammetrie-based systems, the measurement of discrete data points on the surface of an object has become fast and accurate enough for the requirements in industry. The systems provide scattered 3D data points, which in many cases sufficiently describe the model for reconstruction. Any digitising system has different properties and limitations, which have been analysed. It is clear from this analysis, that the surface reconstruction process cannot be solved fully automatically and that a software system should provide the user with tools in order to fix problems due to the scanning system limitations.

There are many applications in industry for these *reverse engineering* problems. The different tasks and possible applications as well as the requirements to be met by a reconstruction systems have been derived in an industrial marketing study. The result is a requirement catalogue, which is the basis for the conception of a software system in order to process probed data, which has been worked out in this thesis.

Most of the known applications are not based on discrete data points but on continuous surfaces such as triangular meshes or spline surfaces. Applications such as copy milling, stereolithographie, visualisation, finite element analysis and quality analysis can be applied to triangular meshes. Also, there are systems for spline surface

generation available, based on triangular meshes. A software system for *reverse engineering* should therefore be able to generate triangular meshes from digitised data. The automatic generation of these meshes is the most difficult problem to be solved in this reconstruction process. We therefore presented an approach in order to solve this problem quickly, efficiently and reliably.

First, an appropriate data structure has been derived for a fast and efficient representation of triangular meshes. The representation is based on the boundary representation of solid models. Restricting the representation with the special properties of triangular meshes leads to a structure, which is efficient and allows fast and easy processing. The data structure can be extended to progressive meshes. This data structure is a continuous resolution representation and allows the display of the surface with a high number of triangles, when the object is near to the viewer and a lower number while the object recedes.

The main problem to be solved within the computer is the generation of meshes from digitised data. Therefore an algorithm has been developed and presented in order to solve the problem quickly, efficiently and reliably. The approach is based on the principle of *multiple view combination* which has also been discussed in the literature. Any digitising system, currently used in industry, produces data in *multiple views*, which means that single scans are related to a fixed direction of view of the scanning system. Any digitised view is first triangulated in the viewing plane using a 2D step discontinuity triangulation approach. This approach covers corner detection and is very fast and easy to implement. The different meshes, generated for each single view are then combined using a *limitation* and *gap bridging* approach. The presented algorithm has a number of desirable properties. The most important properties are the reliability and robustness. This will lead to the reconstruction of simple and complex objects from different scanning devices. The algorithm is fast in terms of computing time and effective in terms of the required computing storage due to the sequential processing of views, such that even large data sets with many million data points can be processed. The resulting mesh only consists of data points, thus no further *geometry* has to be introduced. It is guaranteed, that all data points have been considered in the reconstruction process and that the resulting surface approximates the data points within a given tolerance.

A simple mesh optimisation scheme based on edge swaps has been introduced in

order to improve the quality of the reconstructed surface in terms of smoothness. Edge swap operations are applied, if the curvature of the mesh can be improved. The algorithm reorganises the resulting mesh, such that small radii and sharp corners are represented more precisely and smoother than without optimisation.

Meshes, generated with the presented multiple view combination approach consist of very high numbers of triangles and vertices. In order to process these meshes quickly and efficiently, a triangle decimation approach is presented, which minimises the number of triangles, while preserving the original shape of the surface. Vertices are removed from the mesh in a *best first* sequence. The present approach guarantees, that the distance of the resulting mesh towards the original vertices is less than a user-specified tolerance. The decimation functional guarantees a high quality surface, while still a high number of vertices is filtered.

9.2 Future work

The algorithms in this work are the basic functions for generating surfaces from digitised data. The multiple view combination approach with additional mesh optimisation and triangle decimation leads to high quality surfaces. Still, there are areas and problems, which cannot be reconstructed in an automatic process. These problems have been analysed and functionality has been proposed in order to solve them. Areas, where no data could be achieved have to be reconstructed manually and integrated into the model. Difficult geometry such as small radii or sharp corners are not represented in the data points. There are many open problems, resulting from the limitations of the scanning devices. The aim is to provide the user with tools in order to reconstruct and integrate this features into the object manually.

There are also some problems still remaining within the present approach. The combination approach can be improved in terms of quality. Triangles with *low* quality in overlapping regions are removed from one mesh and triangles from other meshes are inserted. These triangle might be of lower quality such that insertion is not desirable. There are approaches of determining, wheather a triangle is overlapped by others of higher quality, which would overcome these problems. The basic problem is to find a balance between removing triangles and generating gaps, which leads to large computing times.

The most important remaining task is to develop a polygonisation method that is able to process unstructured data points such as point clouds or equivalent. New handheld scanning devices produce data points, which do not have one specific scanning direction. The scanner sweeps across the surface in some sort of paintbrush manner. Data of this type cannot be processed using the present approach since one fixed scanner orientation is required. There are ideas to subdivide this type of data into areas with almost equal sensor orientation. These areas could then be triangulated separately in the common plane and combined with the presented approach. In order to process all kinds of digitised data, this problem needs to be solved.

Finally, not all applications are based on triangular meshes. The storage required for highly detailed meshes is relatively high. The surface is piecewise linear and higher order continuity can hardly be derived. This problems can be overcome with spline based surface models, which lead to a tangential and even curvature continuous representation of the model. The required storage can be reduced and the representation is a today's standard in almost any software system for CAD/CAM.

Appendix A

About Tebis AG

The software, developed in this work is implemented in the Tebis CAD/CAM System. A copy of the Tebis CAD/CAM software package can be found on the last page of this thesis. For further information or a software evaluation licence please contact me at Tebis.

Tebis AG
Lademannbogen 128
22339 Hamburg
Germany
Fon: Germany + (0)40 / 538923-0
Fax: Germany + (0)40 / 538923-20
Mail: schinke@tebis.de

Tebis AG is a software company that develops CAD/CAM systems for tool, die and mould manufacturing. Tebis supplies turnkey installations and also provides a full software service including installation, training and hotline support. Tebis CAD/CAM specialists are available as well to provide their expertise to solve industrial application questions.

Tebis customers come predominantly from the automobile and aircraft industry as well as their subcontractors, predominantly from the styling, model making, tool and mould making manufacturing sectors. In the meantime, Tebis has more than 3000 systems installed at more than 1300 sites world-wide. In addition, companies in the household applience and sanitary installation fields are working with CAD/CAM

software from Tebis AG as are firms in medicine and dentistry. One important prerequisite for a broad installation base is that both the CAD and the CAM modules be easy to learn and operate as well as the high quality of the components that Tebis customers can manufacture on their milling centers. Not least the high CAD/CAM competence of the application engineers plays an ever more decisive role in optimising existing process lines by integrating Tebis CAD/CAM systems.

Tebis software also include fields such as quality control (CAQ) and workshop-oriented programming (WOP). All products are based on the same database, the same release status and the same user interface. The object-oriented system environment is used to make operation comfortable, logical and reliable.

Bibliography

[Ash] I. Ashdown. Boundary representation bibliography. ledalite.com/pub/b-rep97.bib.

[B.74] Baumgart B. *Geometric Modelling for Computer Vision.* PhD thesis, Stanford University, 1974. Also as Technical Report CS-463.

[Blaa] J.M. Blackledge. Digital image processing, lecture notes. DeMontfort University, Science and Engineering Research Centre.

[Blab] J.M. Blackledge. Digital signal processing, lecture notes. DeMontfort University, Science and Engineering Research Centre.

[Blac] J.M. Blackledge. Signal analysis, lecture notes. DeMontfort University, Science and Engineering Research Centre.

[BM92] P.J. Besl and N.D. McKay. A method for registration of 3d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[CKS98a] S. Campagna, L. Kobbelt, and H.P. Seidel. Efficient decimation of complex triangle meshes. *Technical Report 3/98*, 1998. www9.informatik.uni-erlangen.de/Persons/Campagna/papers/index.html.

[CKS98b] S. Campagna, L. Kobbelt, and H.P. Seidel. A general framework for mesh decimation. *Proceedings of Graphics Interface*, pages 43–50, 1998. www9.informatik.uni-erlangen.de/Persons/Campagna/papers/index.html.

[CSYL88] B.K. Choi, H.Y. Shin, Y.I. Yoon, and J.W. Lee. Triangulation of scattered data in 3d space. *Computer Aided Design*, 20(5):239–248, 1988.

[Cur97] B.J. Curless. *New Methods for Surface Reconstruction from Range Images.* PhD thesis, Stanford University, 1997. graphics.stanford.edu/.

[dBvKOS91] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry.* Springer Verlag, 1991.

[DLR90] N. Dyn, D. Levin, and A. Rippa. Data dependent triangulations for piecewise linear interpolation. *IMA Journal of Numerical Analysis*, 10:137–154, 1990.

[EM94] H. Edelsbrunner and E.P. Mücke. Three dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72, 1994.

[GS85] L. Guibas and J. Stolfi. Primitives for the manupulation of general subdivisions and the computation of voronoi diagrams. *ACM Transactions on Graphics*, 4(2), 1985.

[HDD+92] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction of unorganised points. *SIGGRAPH Proceedings*, pages 71–78, 1992.

[HDD+93] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimisation. *SIGGRAPH Proceedings*, 27:19–26, 1993.

[Hil95] A. Hilton. On reliable surface reconstruction from multiple range images. *Technical report VSSP-TR-5/95*, 1995.

[Hoc97] Hans-Jürgen Hochfeldt. Qualitätskriterien für cad flächen. *CAD-CAM Report*, 4:40–46, 1997.

[Hop96] H. Hoppe. Progressive meshes. *Computer Graphics Proceedings*, pages 99–108, 1996.

[Hop97] H. Hoppe. View-dependent refinement of progressive meshes. *Computer Graphics Proceedings*, pages 189–198, 1997.

[Hor87] B.K.P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America*, 4(4):629–642, 1987.

[HR98] Hans Hagen and Dieter Roller. Cad-freiformflächen, qualitätsanalyse. *CAD-CAM Report*, 6:92–98, 1998.

[HSIW96a] A. Hilton, A.J. Stoddart, J. Illingworth, and T. Windeatt. Building 3d graphical models of complex objects. *Technical report*, 1996.

[HSIW96b] A. Hilton, A.J. Stoddart, J. Illingworth, and T. Windeatt. Marching triangles: range image fusion for complex object modelling. *International conference on image processing*, 1996.

[HSIW96c] A. Hilton, A.J. Stoddart, J. Illingworth, and T. Windeatt. Reliable surface reconstruction from multiple range images. *In 4th European Conference on Computer Vision*, pages 117–126, 1996.

[Kar97] Stefan Karbacher. *Rekonstruktion und Modellierung von Flächen aus Tiefenbildern.* PhD thesis, Friedrich-Alexander-Universität Erlangen, 1997.

[KLS96] R. Klein, G. Liebich, and W. Straßer. Mesh reduction with error control. *Technical Report*, 1996.

[Kob97] L. Kobbelt. Discrete fairing. *Proceedings of the Seventh IMA Conference on the Mathematics of Surfaces*, pages 101–131, 1997. www9.informatik.uni-erlangen.de/Persons/Kobbelt/publist.html.

[Kob98a] L. Kobbelt. Polygonal meshes implemented. *to be submitted*, 1998. www9.informatik.uni-erlangen.de/Persons/Kobbelt/publist.html.

[Kob98b] L. Kobbelt. Variational design with parametric meshes of arbitrary topology. *submitted for publication*, 1998. www9.informatik.uni-erlangen.de/Persons/Kobbelt/publist.html.

[KS99] A. Khodakovsky and P. Schröder. Fine level feature editing for subdivision surfaces. *Proceedings of ACM Solid Modeling*, 1999. www.cs.caltech.edu/ ps.

[Law77] C.L. Lawson. Software for c^1 interpolation. In *Mathematical Software III*, pages 161–194. J.R. Rice, eds., Academic Press, 1977.

[LC87] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer and Graphics*, 21(4), 1987.

[Män88] M. Mäntylä. *An Introduction to Solid Modeling.* Computer Science Press, 1988.

[Nie94] Mads Nielson. Surface reconstruction: Gncs and mfas. forschungsbericht 2353. *Institute National de Recherche en Informatique et en Automatique*, 1994. ftp.inria.fr/INRIA/publication/RR/RR-2353.ps.gz.

[NK97] P.J. Neugebauer and K. Klein. Adaptive triangulation of objects reconstructed from multiple range images. *IEEE Visualisation*, pages 20–24, 1997. Late breaking hot topics.

[NK98] P.J. Neugebauer and K. Klein. Adaptive triangulierung komplexer oberflächen. In *ABW-Workshop*, 1998.

[Now99] G.D. Nowottny. *Netzerzeugun durch Gebietszerlegung und duale Graphenmethode.* PhD thesis, Universität Stuttgart, 1999.

[PS85] F. Preparata and M. Shamos. *Computational Geometry.* Springer, 1985.

[RST94] M. Rutishauser, M. Stricker, and M. Trobina. Merging range images from multiple range images. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 573–580, 1994.

[Sch00] J Schuster. *A CAD/CAM concept for the ...* PhD thesis, DeMontfort University, 2000. To be submitted.

[Sed92] Robert Sedgewick. *Algorithmen.* Addison Wesley, 1992.

[SL95a] Marc Soucy and Denis Laurendeau. A dynamic integration algorithm to model surfaces from multiple range images. *Machine Vision and Applications*, 8:53–62, 1995.

[SL95b] Marc Soucy and Denis Laurendeau. A general surface approach to the integration of a set of range views. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 14(4):348–353, 1995.

[TL95] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. *SIGGRAPH Proceedings*, 1995.

[VMC97] T. Varady, R. Martin, and J. Cox. Reverse engineering of geometric models - an introduction. *Computer-Aided Design*, 29:255–268, 1997.